Titre: Title:	Etude et validation de modèles neuronaux de mémoires corrélatives adressables par leur contenu
Auteur: Author:	Jong-Mo Allegraud
Date:	2002
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Allegraud, J.-M. (2002). Etude et validation de modèles neuronaux de mémoires corrélatives adressables par leur contenu [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/6981/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/6981/
Directeurs de recherche: Advisors:	
Programme: Program:	Non spécifié

NOTE TO USERS

UNIVERSITÉ DE MONTRÉAL

ÉTUDE ET VALIDATION DE MODÈLES NEURONAUX DE MÉMOIRES CORRÉLATIVES ADRESSABLES PAR LEUR CONTENU

JONG-MO ALLEGRAUD
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE ÉLECTRIQUE)
NOVEMBRE 2002

UNIVERSITÉ DE MONTRÉAL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

ÉTUDE ET VALIDATION DE MODÈLES NEURONAUX DE MÉMOIRES CORRÉLATIVES ADRESSABLES PAR LEUR CONTENU

présenté par : ALLEGRAUD Jong-Mo
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. NERGUIZIAN Chahé, M.Sc.A., président
M. BRAULT Jean-Jules, Ph.D., membre et directeur de recherche
M. SAWAN Mohamad, Ph.D., membre

NOTE TO USERS

RÉSUMÉ

Depuis une vingtaine d'années, les systèmes à base de réseaux de neurones connaissent un regain de popularité. On les utilise notamment dans la plupart des systèmes de reconnaissance vocale et de reconnaissance de formes. En effet, les réseaux de neurones ont la faculté de retrouver rapidement les données qu'ils ont enregistrées. Cette propriété semble intéressante pour développer des « Content Addressable Memories » ou CAM à base de réseaux de neurones. Les CAM sont des mémoires particulières dans lesquelles la donnée détermine l'endroit où elle est stockée. Il existe sur le marché des CAM très performantes, mais elles sont également très dispendieuses en raison des méthodes d'implantation utilisées. Pour cette raison, les CAM sont utilisées seulement dans les systèmes où la vitesse de récupération des enregistrements est critique. Ainsi, la recherche effectuée a pour objectif d'étudier des modèles de CAM neuronales et d'évaluer qualitativement leur intérêt en terme de coûts et de performances.

Toutefois, pour obtenir des CAM neuronales fiables, certaines caractéristiques des réseaux de neurones doivent être maîtrisées. En particulier, la capacité des réseaux de neurones à inférer une réponse en présence d'une entrée inconnue, si elle est avantageuse dans de nombreuses applications, est problématique pour le développement de CAM neuronales. En effet, cette propriété a pour conséquence une incapacité du réseau de neurones à différencier les informations connues des informations inconnues. Dans ce contexte, le développement d'une CAM neuronale nécessite une modification des modèles existants.

La recherche consiste donc, dans un premier temps, à proposer et à valider des modèles de CAM neuronales. Pour valider un modèle, il faut vérifier que la CAM neuronale ne commette aucune erreur ni avec les vecteurs enregistrés (erreur de type I) ni avec les

vecteurs non enregistrés (erreur de type II). Pour les modèles validés, on procède à une deuxième étape qui doit déterminer les ressources mémoires et le nombre d'opérations nécessaires. Cette deuxième étape permettra de donner une estimation qualitative du coût et des performances des CAM neuronales étudiées. Bien que plusieurs modèles de réseaux de neurones soient envisageables pour développer une CAM neuronale, la recherche se limitera à l'étude du modèle des « Correlation Matrix Memories » ou CMM qui est un modèle de mémoire distribuée à base de réseaux de neurones.

On peut a priori utiliser les CMM soit comme une CAM neuronale à part entière, soit pour effectuer un prétraitement dans la CAM neuronale. Néanmoins, les simulations effectuées montrent que les CMM ne peuvent être utilisées seules pour effectuer un traitement sans erreur : si l'orthogonalisation des vecteurs garantit l'absence d'erreurs de type I, elle ne peut assurer l'absence d'erreurs de type II. Toutefois, les CMM permettent un prétraitement efficace : avec des vecteurs de 128 bits, le prétraitement réduit de plus de 99% le nombre de tests à effectuer. De plus, compte tenu des propriétés des CMM, ce prétraitement doit être encore plus efficace avec des vecteurs d'entrée de plus grande taille.

D'autre part, dans le modèle des CMM, il faut, idéalement, que les vecteurs d'entrées enregistrés soient orthogonaux si l'on veut pouvoir retrouver les informations enregistrées. Ainsi, deux approches sont étudiées : la première utilise une orthogonalisation classique des vecteurs d'entrées, la deuxième projette les vecteurs à enregistrer dans un espace de grande dimension en utilisant un processus partiellement aléatoire. On constate que les modèles utilisant une orthogonalisation exacte des vecteurs d'entrée sont très exigeants en ressources et présentent un intérêt limité en terme de coût et de performances. En revanche les modèles utilisant la projection des vecteurs d'entrée dans un espace de grande dimension sont peu exigeants en ressources mémoire et en ressources de calculs. Les données récupérées indiquent que, pour des entrées de 128 bits, les CAM neuronales peuvent présélectionner un vecteur parmi 112.

Ce prétraitement nécessite seulement 69 bits de mémoire et 140 opérations logiques par vecteur enregistré.

D'autre part, les CAM neuronales à base de CMM permettent de travailler avec des vecteurs d'entrée de très grande taille : cette caractéristique est particulièrement intéressante car le traitement des vecteurs de grande taille est très coûteux dans les CAM « traditionnelles ». Ainsi, comme les CAM neuronales peuvent traiter les vecteurs de grande taille efficacement, elles semblent particulièrement adaptées au filtrage de paquets TCP/IP dans les systèmes coupe-feu (Firewall). En effet, ces systèmes utilisent des règles d'au moins 128 bits de long.

ABSTRACT

During the past twenty years, neural networks based systems have been regaining popularity. Such systems are widely used for voice and pattern recognition applications, since they are able to retrieve quickly the data they have previously stored. This property is of big interest to develop neural based "Content Addressable Memories" (CAMs). CAMs are special memories for which the data determines the memory location where it will be stored. CAMs available in the marketplace are very efficient but also very expensive due to their architecture. Thus, CAMs are only used in applications that demand very fast data retrieval. In this context, this research aims at finding valid neural CAM models and at giving a first evaluation of their interest as to cost and performance.

Even though neural based methods are an efficient way of processing data; it is difficult to get reliable neural based CAMs and some neural networks specific features must first be controlled. The control of neural networks inference capabilities that prevent systems to differentiate known data from unknown data is the main problem that needs to be solved. In such a situation, existing neural networks models must be modified to fit their use in a neural CAM.

The research consists first in proposing and validating neural CAM models. For a model to be validated, the neural CAM has to make no mistake whether it is with vectors previously recorded (error of type I) or with unknown vectors (error of type II). For each validated model, memory and processing requirements are computed, which enables to get a first estimation of cost and performance for these systems. Even if several neural models could be studied, the research focuses on the "Correlation Matrix Memory" (CMM) model, which is a neural-based distributed-memory model.

CMMs can be used either as a stand-alone neural CAM or as a pre-processor in the neural CAM. Nevertheless, the simulations show that using a CMM alone to design a neural CAM is not reliable: the orthogonalization process insures there will not be any error of type I, it cannot, however, prevent from having errors of type II. Nevertheless, CMMs can achieve a very efficient pre-processing: with 128-bits input vectors, CMMs' pre-processing reduces the amount of tests to be done by over 99%. Moreover, this pre-processing task should be all the more efficient, as the input size increases.

Besides in the CMM model, recorded input vectors should be ideally orthogonal for a correct retrieval of the stored data. Consequently, two approaches have been studied: the first one considers a traditional orthogonalization of input vectors and the second one, a projection of the input vectors into a high-dimension space through a partially random transformation. On the one hand, models using a traditional orthogonalization of the input vectors are very demanding and do not meet the expected goals as to cost and performance. On the other hand, methods using the projection of the input vectors into a high-dimension space have very interesting features for both memory and processing requirements. The simulations show that a CMM can pre-select one vector out of 112. This pre-processing task requires only 69 bits and 140 logical operations per recorded vector.

Moreover, whereas "traditional" CAMs are very costly with big input sizes, neural CAMs are all the more efficient as the input size increases. As neural CAMs are efficient with big input sizes, they are well fit for applications such as TCP/IP packet filtering in the firewalls that requires rules of at least 128 bits.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

CAM	:	Content Addressable Memory
CMM	:	Correlation Memory Matrix
DSP	:	Digital Signal Processor
FPGA	:	Field Programmable Gate Array
IP	:	Internet Protocol
IP	:	Intellectual Property
k-NN	:	k-Nearest Neighbours
LUT	:	Look Up Table
NPU	:	Network Processing Unit
RAM	:	Random Access Memory
SDM	:	Sparse Distributed Memory
TCP	:	Transport Control Protocol

LISTE DES ANNEXES

CHAPITRE 1
INTRODUCTION

Le développement des réseaux de neurones artificiels a été inspiré par le fonctionnement du cerveau humain, sorte d'« ordinateur » au fonctionnement très complexe, très non-linéaire et fortement parallèle [HAY98]. Depuis l'invention des réseaux de Hopfield [HOP82] et de l'algorithme de rétro propagation de l'erreur [RUM86], les systèmes à base de réseaux de neurones connaissent un regain de popularité. Aujourd'hui de nombreux systèmes de traitement de la voix ou de traitement de l'image font appel aux réseaux de neurones [FRE01], [DELO99]. En effet, les réseaux de neurones ont la capacité de retrouver rapidement une donnée grâce à un processus de lecture fortement parallèle. En outre, ils ont la capacité d'interpréter les données qu'ils connaissent pour inférer des réponses lorsqu'on leur présente des entrées inconnues. Le but de la recherche est d'utiliser des réseaux de neurones pour développer certains modèles de mémoires.

Plus précisément, on cherche à implanter des réseaux de neurones dans des mémoires destinées aux bases de données pour lesquelles on s'efforce de minimiser le temps de recherche. Il existe actuellement des mémoires dédiées à cette tâche : ce sont les mémoires adressables par leur contenu [DELG99], [DIT00], plus connues sous leur dénomination anglaise de « Content Addressable Memories » ou CAM. Toutefois, les CAM actuelles ne sont utilisées que si le temps de recherche est critique car leur prix est très élevé, comme dans les routeurs à haut débit [MCA93], [LIU01] : pour de nombreuses applications, une approche logicielle est privilégiée. Ainsi, dans le cadre d'un projet industriel, nous avons choisi d'explorer la possibilité d'utiliser des réseaux de neurones pour concevoir des CAM.

1.1 INTÉRÊT DES « CONTENT ADDRESSABLE MEMORIES »

Les CAM bénéficient actuellement d'une très grande popularité avec l'arrivée des connexions internet à haut débit. En effet pour pouvoir gérer des débits de 1 voire plusieurs gigabits par seconde, il est nécessaire de traiter les connexions avec des routeurs capables de traiter des débits de plusieurs gigabits par seconde : dans ce cas il est nécessaire d'accéder aux données des tables de routages avec la plus grande vitesse possible. Ainsi les routeurs haut de gamme utilisent le plus souvent des CAM pour enregistrer les tables de routage.

Dans le domaine de la sécurité informatique, on retrouve un problème assez similaire pour les systèmes coupe-feu ou « firewalls » (Annexe I). Le rôle principal de ces systèmes est de sécuriser le réseau interne en contrôlant les connexions qu'il établit avec l'internet. Si la tâche première du « firewall » est de protéger le réseau interne, on cherche à effectuer le contrôle des connexions sans trop diminuer le débit entre les 2 réseaux. La tâche première d'un système pare-feu est le filtrage de paquets IP (Annexe II) : le « firewall » doit vérifier que tous les paquets respectent les règles de filtrage établies. Dans ce domaine, les CAM ne sont pas utilisées en raison de leur prix trop élevé : si les CAM simples ne requièrent que peu de ressources matérielles, les CAM disponibles sur le marché sont d'une telle complexité que leur prix est élevé.

1.1.1 Vers une définition des CAM

Les CAM ont pour particularité d'être les mémoires les plus performantes pour toutes les tâches d'appariement entre une entrée et une sortie. Ainsi, on peut définir une CAM de la manière suivante :

> «With conventional indexing schemes the data content is used with a hash or index to produce the address location of the data. The address has no real or direct relationship with the information contained in the data. With CAM, the data describes its own storage location. This also means all

> like data will always be found close together in the physical data structure. There is a direct relationship between the information in the data and its location in the physical data store.» [OPE99]

La différence entre les CAM et les autres mémoires se situe au niveau de la méthode d'enregistrement. Dans la plupart des mémoires, telles les RAM, l'adresse mémoire n'a pas de véritable relation avec son contenu. Les CAM, au contraire, établissent un lien entre l'adresse et la donnée : une CAM est une mémoire pour laquelle la donnée détermine l'adresse d'enregistrement. C'est grâce à cette stratégie de stockage que les CAM sont des systèmes très performants.

1.1.2 Une popularité grandissante...

Actuellement, les CAM sont utilisées pour les routeurs qui doivent gérer de gros débits (10 Gbits/s voire plus). Avec le développement de l'internet et l'augmentation des besoins en bande passante, les CAM connaissent une très grande popularité. Ainsi, les recherches effectuées dans ce domaine ainsi que l'augmentation de la demande ont permis une baisse du prix de ces mémoires. Certains, tel John Boyd, chef de projet chez Music Semiconductors Inc[1], évoquent leur utilisation prochaine dans de plus en plus de domaines :

> «CAM can be used for anything that requires some sort of search or translation. Although CAM could be used in other applications, it excels in data internetworking [...] Although the basic memory architecture of CAM has not changed in the last decade, devices have become faster, bigger, and more sophisticated. Conversely, improvements in design and production have allowed manufacturers to reduce the cost to the consumer to make it more attractive. These factors have made CAM increasingly popular in today's layer 2, layer 3, and layer 4 networking applications. CAM will allow designers to meet the demands of bigger, faster, and more sophisticated switching and routing products.»
> http://www.eedesign.com/design/cam/cam.html

[1] Music Semiconductors Inc. est un fabricant de CAM

1.1.3 Mais une démocratisation difficile

L'optimisme de John Boyd n'est toutefois pas partagé par tous. Ainsi Linley Gwennap, consultant en analyse et stratégie pour les technologies réseautiques analyse le marché des CAM de la manière suivante dans EE Times :

> « The reason that CAMs have yet to fulfill their promise is that, down inside, no one really likes them. CAMs are power-hungry and expensive. In some designs, the CAMs cost more than the NPU. » [GWE02]

On peut ajouter à cela que pour le cas particulier des règles d'un système coupe-feu, l'emploi des CAM serait très coûteux car ces règles ont typiquement une longueur d'au moins 128 bits.

1.2 UNE MUTATION DES BESOINS

Les CAM ont deux caractéristiques principales qui sont sa largeur, représentant la taille d'entrée et sa profondeur, nombre d'enregistrements que peut stocker la CAM : cette notion est comparable à la capacité mémoire d'une RAM à la différence que dans une CAM, on ne peut pas organiser l'espace mémoire comme on le désire.

1.2.1 D'un besoin de CAM peu larges mais profondes...

Le rôle du routeur est d'acheminer un paquet IP vers la bonne destination : le routeur va donc déterminer vers où doit être acheminé le paquet IP en fonction de l'adresse IP de destination. Le routeur doit connaître un grand nombre de règles de routages et la CAM utilisée doit ainsi avoir une grande profondeur mais n'a pas besoin d'une largeur très grande.

1.2.2 Vers un besoin de CAM larges et profondes

Cependant, si on désire utiliser des CAM dans des systèmes coupe-feu, on a besoin autant d'une grande largeur, afin de pouvoir mémoriser une règle entière, que d'une grande profondeur.
De plus, l'arrivée progressive de la norme IPv6, conçue afin de faciliter la gestion des paquets IP et l'attribution d'adresses IP, va accentuer les besoins en CAM larges :

- La taille des adresses IP va passer de 32 à 128 bits augmentant ainsi considérablement la largeur de CAM nécessaire au routage des paquets. Ceci aura aussi une très grosse influence sur la taille des règles dans les systèmes pare-feu.
- La taille des numéros de port pourrait passer de 16 à 32 bits dans le protocole TCPv6, ce qui augmenterait encore la taille des règles de « firewall ».

Ainsi cette nouvelle norme implique des contraintes difficiles à respecter en terme de largeur de CAM aussi bien pour le routage que pour l'implantation de règles dans un système coupe-feu. Le passage à la version 6 du protocole IP va donc avoir une grande incidence sur l'industrie des CAM dont la largeur minimale devra passer de 64 bits à 240 voire 300 bits. Et si les fabricants de CAM annoncent de nouvelles CAM capables de supporter de telles largeurs, cela devrait impliquer une augmentation importante de leur prix. Il serait donc intéressant d'étudier de nouvelles façons de concevoir des CAM. Ainsi, nous nous proposons d'explorer une nouvelle approche.

1.3 POSSIBILITÉ DE DÉVELOPPER DES CAM NEURONALES

1.3.1 Motivation d'une telle étude

Il semble possible de développer des méthodes alternatives à base de réseaux de neurones. En effet, par opposition à ces CAM que l'on peut appeler CAM

« traditionnelles », il existe d'autres modèles de CAM : ce sont les CAM neuronales, c'est-à-dire utilisant des réseaux de neurones. Néanmoins, les CAM neuronales existantes sont incapables de différencier les vecteurs qu'elles ont appris, des vecteurs qui leur sont inconnus.

1.3.2 Objectifs visés

Les travaux réalisés ont pour but de déterminer la faisabilité de CAM neuronales dans le but de réaliser un appariement exact. Une fois cette faisabilité établie, le but recherché est d'estimer les potentialités d'une telle architecture en donnant des données quantitatives pour évaluer son coût et ses performances.

1.4 LES ÉTAPES DE LA RECHERCHE

Ainsi la recherche comporte plusieurs étapes qui permettent de déterminer par raffinements successifs des modèles réalisant les objectifs fixés. La première étape consiste à identifier les problèmes inhérents aux réseaux de neurones. Parmi tous les modèles de réseaux de neurones existants, les réseaux de neurones adaptés au problème de recherche de règles ont été identifiés. Par la suite un de ces modèle a été étudié : il s'agit du modèle des matrices de corrélation ou « Correlation Memory Matrices » ou CMM.

1.4.1 Les CAM neuronales : une voie de recherche pertinente ?

L'idée d'utiliser des réseaux de neurones dans le cas d'un appariement exact comporte a priori un avantage et un inconvénient principaux. L'avantage des réseaux de neurones est leur manière de stocker les données qui permet de retrouver rapidement la sortie correspondant à une entrée enregistrée. Mais les réseaux de neurones ont aussi l'inconvénient de ne pas pouvoir différencier le « connu » de l'« inconnu » car ils

cherchent toujours à donner la meilleure sortie possible ou la sortie la plus plausible lorsqu'on leur présente une entrée. Compte tenu de ces deux propriétés, est-il possible de trouver des architectures à base de réseaux de neurones pour un appariement exact ?

1.4.2 Les réseaux de neurones existants : une base sur laquelle s'appuyer ?

Comme il a été mentionné auparavant, il est possible de relier une entrée à l'emplacement mémoire où elle a été enregistrée grâce à des modèles neuronaux. Toutefois, il n'est pas facile d'adapter ce genre de modèles pour créer des CAM et deux stratégies sont envisageables :

- La première consiste à utiliser des réseaux de neurones pour déterminer directement si un vecteur a été enregistré. Dans ce cas, le réseau de neurones s'identifie à la CAM neuronale et la tâche à effectuer équivaut à tester le vecteur d'entrée avec tous les vecteurs enregistrés.
- La deuxième consiste à utiliser les réseaux de neurones pour effectuer un prétraitement dans lequel on sélectionne un sous-ensemble de vecteurs enregistrés. Ce prétraitement doit garantir que ce sous-ensemble contienne toujours le vecteur d'entrée s'il est l'un des vecteurs enregistrés.

1.4.3 A la recherche de modèles valides

Pour chacun des modèles étudiés, il faut effectuer une validation. Cette validation doit garantir que le système ne commet aucune erreur. Fondamentalement, il existe deux types d'erreurs :

- Les erreurs survenant avec des vecteurs enregistrés qui sont des erreurs de non détection des vecteurs enregistrés. On appellera ces erreurs, erreurs de type I.
- Les erreurs survenant avec des vecteurs non enregistrés ou erreurs de fausse détection. On appellera ces erreurs, erreurs de type II.

L'identification des erreurs détectées permet d'apporter des modifications pertinentes au modèle étudié. On modifie donc le modèle jusqu'à obtention d'un modèle valide ou jusqu'à ce que l'on constate .que le modèle étudié ne peut être validé.

Lorsque les modèles valides sont identifiés, il faut évaluer pour chacun le coût et les performances. Pour cela, on détermine les ressources mémoire et le nombre d'opérations logiques nécessaires pour implanter la CAM. Toutefois, la discussion sur le coût et les performances des modèles de mémoire développés ne peut être que qualitative car elles sont fortement dépendantes du choix d'implantation.

CHAPITRE 2

INTÉRÊT DE DÉVELOPPER UNE CAM NEURONALE

2.1 INTRODUCTION

Actuellement, les seules CAM existantes sur le marché sont les CAM « traditionnelles » à base de logique pure. Ces CAM sont très dispendieuses : en raison de toutes les fonctionnalités qu'elles offrent, leur conception devient très complexe. De plus, il est intéressant, d'un point de vue théorique, d'étudier les possibilités qu'offrent les réseaux de neurones pour concevoir de tels systèmes : une approche neuronale peut permettre d'offrir une alternative aux CAM « traditionnelles ». Pour cela, il faut tout d'abord identifier les fonctions fondamentales d'une CAM, c'est-à-dire les fonctions indépendamment de l'architecture envisagée. Ensuite, une étude des CAM « traditionnelles » sera faite afin de mieux comprendre leur principe de fonctionnement. A partir de cette base de connaissance, il sera possible de réfléchir aux possibilités d'implanter une CAM utilisant des réseaux de neurones. Enfin, une discussion sera faite sur les objectifs que l'architecture proposée doit réaliser pour offrir une alternative qui soit à la fois performante et économique.

2.2 LES FONCTIONS FONDAMENTALES DE LA CAM

Bien que les CAM « traditionnelles » et les CAM neuronales ne soient pas conçues avec une même approche, elles possèdent des fonctions communes. En effet, comme leur rôle est identique, les deux architectures doivent remplir les mêmes fonctions générales (Figure 2.1) :

- La première fonction à remplir est une fonction de prétraitement qui consiste à déterminer ce que l'on cherche à apparier. Cette fonction est assez différente dans le cas des CAM traditionnelles et dans celui des CAM neuronales. Dans une CAM « traditionnelle », le prétraitement consiste à isoler la partie à apparier sur tous les enregistrements alors que dans une CAM neuronale, le prétraitement doit isoler les enregistrements sur lesquels doit se faire l'appariement.
- La seconde fonction est la fonction d'appariement plus connue sous le terme anglais de « matching » qui permet de comparer l'entrée aux enregistrements pour déterminer si le vecteur présenté en entrée est présent dans la mémoire.

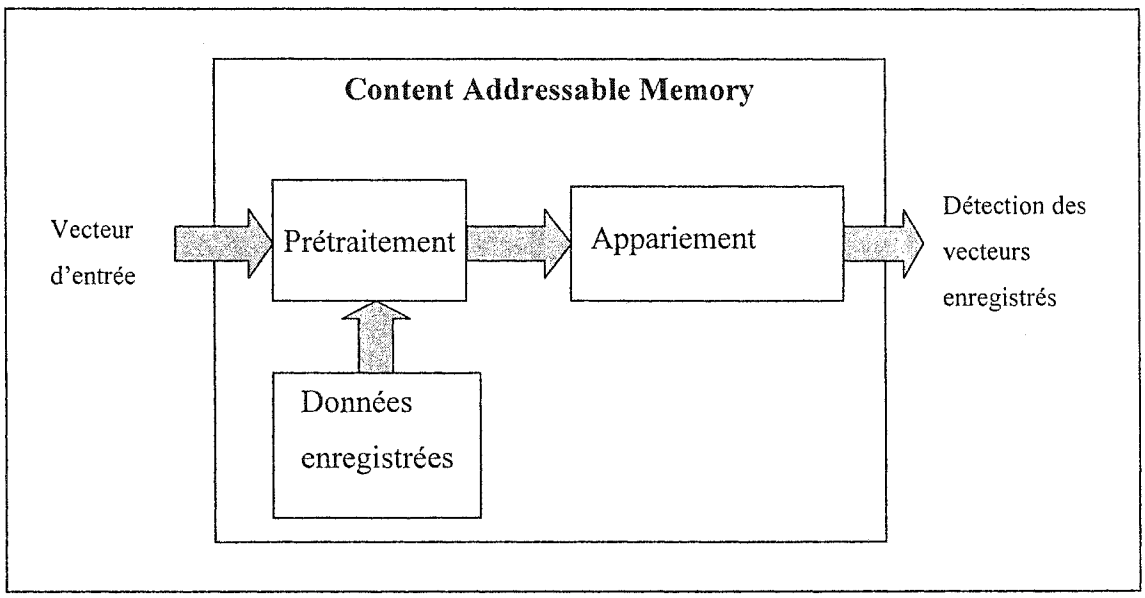


Figure 2.1 : Fonctions fondamentales d'une CAM

2.3 LES CAM « TRADITIONNELLES »

Les CAM commercialisées actuellement sont des systèmes très complexes en raison des fonctionnalités qu'elles doivent offrir. Dans ces systèmes le traitement des données est totalement parallèle et ainsi, chaque cellule de mémoire doit posséder sa propre fonction

de prétraitement et d'appariement. Dans les CAM « traditionnelles », chaque cellule de mémoire correspond à un enregistrement et peut fournir un signal indiquant s'il y a appariement entre la donnée en entrée et la donnée qu'elle a stockée. On distingue les CAM binaires des CAM ternaires : les CAM binaires ne peuvent enregistrer que des vecteurs binaires alors que les CAM ternaires acceptent 3 états '0', '1' et «don't care», couramment noté 'x'. En effet, comme ces mémoires sont très utilisées pour le routage, l'état 'x' permet de déterminer des règles de routage pour des sous-ensembles d'adresses. Le but de cette section étant de comprendre le principe de fonctionnement d'une CAM, seul sera décrit le fonctionnement d'une CAM binaire élémentaire.

2.3.1 Fonction de prétraitement

Dans le cas des CAM « traditionnelles », ce sont les mots enregistrés dans la CAM qui sont prétraités : le prétraitement consiste à sélectionner par masquage la partie des données enregistrées à comparer avec le vecteur d'entrée. Cette opération est très simple et très rapide mais doit se faire au niveau de chacune des cellules de mémoire.

2.3.2 Fonction d'appariement

Dans le cas d'une adresse binaire, on peut très facilement comprendre le principe de fonctionnement grâce à de simples portes logiques "ET" (Figure 2.2). Toutefois, les cellules mémoires des CAM sont, dans la réalité, beaucoup plus complexes. Mais l'approche permet de donner une estimation du nombre de porte logiques nécessaires pour concevoir une CAM. Si la CAM peut enregistrer n vecteurs de s bits, il faut $(n-1)*s$ portes "ET". Si on suppose qu'il y a statistiquement autant de '0' que de '1' dans les vecteurs à enregistrer, on a $(n-1)*s/2$ portes "NON".

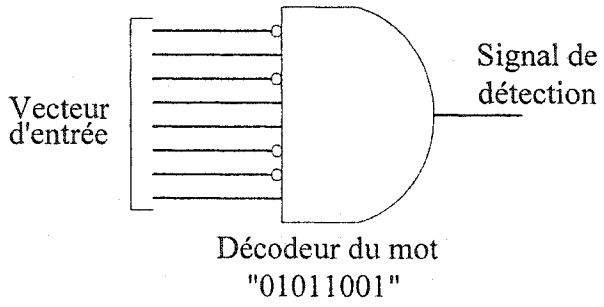


Figure 2.2 : Détecteur d'adresse 8 bits

Bien que les principes élémentaires de fonctionnement des CAM « traditionnelles » semblent très simples, l'architecture globale est coûteuse (Tableau 2.1) car de nombreuses fonctionnalités supplémentaires doivent être implantées, telle la possibilité d'utiliser la CAM comme une RAM.

2.4 LES CAM NEURONALES

A partir du constat fait sur l'architecture des CAM « traditionnelles », on va chercher à créer des CAM neuronales qui puissent réaliser un contrôle pour un ensemble d'enregistrement afin d'économiser des ressources. Parmi les modèles neuronaux existants, il y a plusieurs architectures considérées comme étant des CAM : dans son livre « Introduction to Natural Computation », Ballard [BAL99] distingue les Correlation Matrix Memories (CMM), les Sparse Distributed Memories (SDM) et les mémoires de Hopfield. Contrairement aux CAM «traditionnelles», les CAM neuronales ont pour base une approche connexionniste et cherchent à imiter le comportement des neurones biologiques qui ont la faculté de retrouver rapidement les données qu'ils ont enregistrées. En effet c'est grâce à cette rapidité à retrouver des données que les êtres vivants sont des systèmes très performants pour des tâches telles la reconnaissance de visage ou la reconnaissance des sons.

Dans l'approche neuronale proposée, le réseau de neurones se chargera du prétraitement et un circuit constitué de logique pure se chargera de l'appariement.

2.4.1 La fonction de prétraitement

Contrairement à une CAM « traditionnelle », le prétraitement dans un réseau de neurones est réalisé pour un ensemble d'enregistrements. Le réseau de neurones effectue un pré traitement afin de réduire considérablement la taille de l'espace de recherche : le réseau de neurone permet de déterminer un ensemble d'adresses possibles et doit assurer en outre que, si la donnée a été enregistrée, elle l'a été à l'une des adresses sélectionnées. La difficulté de cette approche est de réussir à garantir que le système se comporte comme désiré dans tous ses états possibles, avec toutes les entrées possibles. Dans le cas d'une entrée de 4 bits, il y a 696 états possibles pour le système (Annexe III) et 16 vecteurs d'entrée possibles : il y a donc 696*16=11136 tests à effectuer. Si pour une entrée de 4 bits, un test exhaustif du système est possible, il est impossible pour une entrée de 8 bits : pour un tel système, il y a $1,354*10^{13}$ états possibles et 256 vecteurs d'entrée possibles. Par conséquent, il faudrait effectuer $1,354*10^{13}*256=2,796*10^{15}$ tests, pour réaliser une vérification exhaustive du système, soit des millions de milliards de tests.

2.4.2 La fonction d'appariement

La fonction d'appariement va permettre de s'assurer de l'existence ou non d'une donnée en mémoire. Cette fonction est réalisée sans utiliser de réseaux de neurones mais avec de la logique pure. Suivant la méthode utilisée, l'appariement ne se fait pas de la même façon. Il peut se faire sur un ou plusieurs candidats sélectionnés par le réseau de neurones.

2.5 UN COMPROMIS COÛT-PERFORMANCE

2.5.1 Les principaux fabricants de CAM « traditionnelles »

Les fabricants de CAM sont assez nombreux : on distingue les fabricants spécialisés dans la fabrication de CAM et les fabricants de FPGA qui vendent des propriétés intellectuelles (*IP*) qui se présentent sous forme de code source qui permet d'implanter une fonction sur un FPGA. Les principaux fabricants de CAM sont Cypress, IDT, Kawasaki, Mosaid et Sibercore.

Tableau 2.1 : Les principaux fabricants de CAM

Marque	Référence	Largeur (bits)	Profondeur (milliers d'enregistrements)	Vitesse (millions de recherches/s)	Prix unitaire ($ US)
Cypress	CYNSE70064A	136 272	16 8	66 33	120 à 270
IDT	75K62100	72	128	100	400 à 560
Kawasaki	KCAM9A	144 288	64 32	100 50	NC
Mosaid	DC18288	144 288	128 64	100 100	NC
Sibercore	SCT 2000	144 288	16 8	100 100	NC

On peut noter que la plupart des FPGA tels les Virtex de Xilinx et APEX II d'Altera fournissent des IP permettant d'implanter des CAM. Toutefois, le prix de tels systèmes se chiffre en milliers de dollars.

2.5.2 Un coût élevé

Les CAM ne sont pas utilisées dans tous les domaines où elles pourraient apporter une nette amélioration des performances. La raison principale de ce désintérêt est leur coût élevé : la plupart du temps, leur intégration n'est pas rentable. Comme il est mentionné dans le Tableau 2.1, le prix unitaire de tels systèmes se chiffre en centaines de dollars en raison des fonctions additionnelles qu'elles doivent remplir tel le fonctionnement en tant que RAM. On ne va les intégrer que dans les systèmes où la vitesse de traitement est un facteur critique. La parallélisation totale évoquée dans la section 2.3 a pour conséquence que leur taille physique et leur coût sont directement proportionnels à leur largeur et à leur profondeur [MCA93].

2.5.3 Des contraintes de plus en plus exigeantes

Vue la taille des règles des systèmes pare-feu qui est actuellement d'au moins 128 bits, l'utilisation d'une CAM dans ce domaine est déjà peu intéressante. De plus, avec la venue de la norme IPv6, l'utilisation d'une CAM semble peu envisageable. Ainsi, pour pouvoir intégrer des CAM dans des systèmes pare-feu matériels, il faut en diminuer les coûts, quitte à perdre un peu en matière de performances. L'objectif visé dans cette recherche est de trouver un modèle de mémoire alternative moins coûteux mais performant. On peut représenter cet objectif dans un plan coût-performance (Figure 2.3).

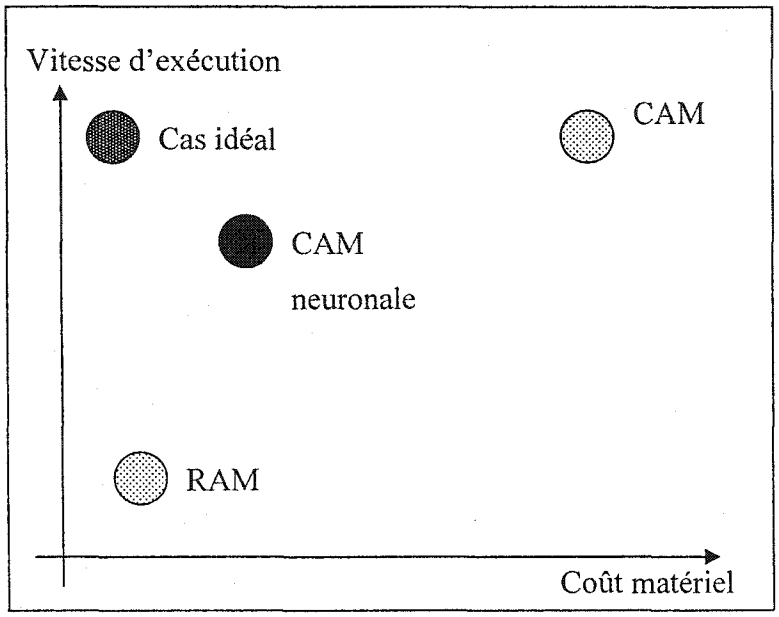


Figure 2.3 : Le compromis coût-performance recherché

2.6 CONCLUSION

Dans ce chapitre, on a proposé une définition des fonctions essentielles d'une CAM. On constate que les CAM ont une fonction de prétraitement qui prépare les données pour la phase d'appariement. La phase d'appariement est une étape très rapide. On constate que, dans les CAM « traditionnelles », des fonctionnalités secondaires inutiles doivent être mises en place, ce qui rend complexe la réalisation des cellules de mémoire. Comme on peut le constater sur le Tableau 2.1, la conséquence de cette complexité est un prix de vente très élevé. Ainsi, les modèles de CAM neuronales envisagés doivent optimiser l'utilisation des ressources pour le stockage et le traitement des données enregistrées.

CHAPITRE 3

RÉSEAUX DE NEURONES FONDAMENTAUX UTILISÉS

3.1 INTRODUCTION

Il existe de nombreux modèles de réseaux de neurones et il faut réussir à sélectionner parmi eux ceux qui peuvent permettre de bonnes performances. Comme il a été vu précédemment (section 2.4), il existe plusieurs modèles de CAM neuronales. Parmi ces modèles, deux sont particulièrement intéressants car ils ont déjà été implantés à un niveau matériel : ce sont les modèles des SDM (Annexe IV) et des CMM. Bien qu'une adaptation du modèle des SDM, à l'instar du modèle de CMM développé, offre de bonnes perspectives en terme de coût et de performances, seule une étude sur les CMM a été menée à terme, faute de temps.

Afin de progresser dans le choix d'un modèle, une étude des spécificités des réseaux de neurones a été menée. Une telle étude permet de mettre en lumière les difficultés à affronter pour adapter un modèle neuronal à un problème de détection de règles. Par l'intermédiaire du modèle des k « nearest neighbors » ou k-NN, on illustrera les caractéristiques des réseaux de neurones utiles à la détection de règles. Enfin, on donnera une description du modèle utilisé par la suite : le modèle des CMM.

3.2 SPÉCIFICITÉ D'UNE APPROCHE NEURONALE

La caractéristique principale des réseaux de neurones est leur faculté d'apprentissage. Dans le contexte de la détection de règles, l'approche neuronale va permettre de traiter

un ensemble de règles, ce qui n'est pas le cas dans l'approche traditionnelle où l'on traite les règles indépendamment les unes des autres.

3.2.1 Principe de fonctionnement d'un réseau de neurones

3.2.1.1 Le modèle de McCulloch-Pitts

Dans un des modèles les plus couramment utilisés, le modèle de McCulloch-Pitts [MCC43], un neurone est modélisé par N entrées –les synapses-, un seuil d'activation, un sommateur, une fonction d'activation et une sortie. Les entrées sont pondérées par des poids synaptiques ajustables : c'est grâce à cette possibilité de modifier ses poids synaptiques qu'un neurone a la faculté d'apprendre.

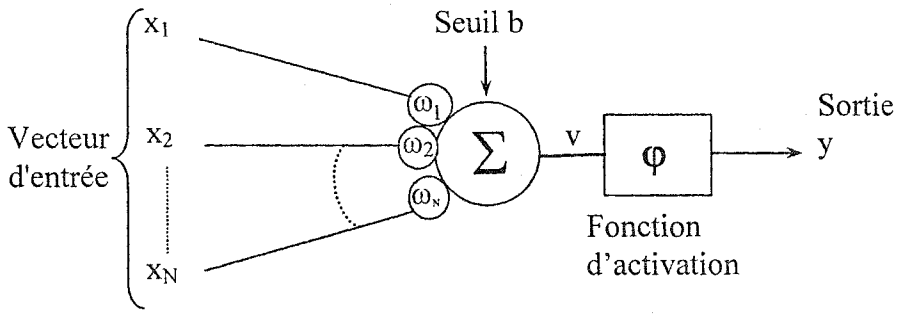


Figure 3.1: Le modèle de neurone de McCulloch-Pitts

Dans le modèle présenté sur la Figure 3.1, on a les relations :

$$v = \sum_{i=1}^{N} \omega_i x_i + b \tag{3.1}$$

$$y = \varphi(v) = \varphi\left(\sum_{i=1}^{N} \omega_i x_i + b \right) \tag{3.2}$$

La fonction d'activation est la plupart du temps une fonction « Heavyside », une sigmoïde ou une tangente hyperbolique. La sigmoïde est une fonction mathématique qui a pour équation :

$$\varphi(v) = \frac{1}{1+e^{-a*v}} \; a \in \Re \tag{3.3}$$

L'allure de la tangente hyperbolique est identique à celle de la sigmoïde mais elle varie entre –1 et 1.

3.2.1.2 Des neurones en réseau

À partir de cette unité de base, on peut construire des réseaux de neurones[2] capables d'accomplir des tâches complexes. De manière générale, les réseaux de neurones ont la capacité d'interpoler des fonctions même dans de grandes dimensions ou de classer des échantillons inconnus grâce aux connaissances qu'ils ont acquises : on parle de généralisation. Cependant, il est nécessaire d'entraîner un réseau de neurones –période d'apprentissage- avant qu'il se comporte de la manière souhaitée. Cet apprentissage consiste en une correction des poids synaptiques pendant la période d'entraînement. Dans la plupart des cas, cette correction se fait à l'aide d'une fonction de coût qu'il faut déterminer puis minimiser en modifiant les poids synaptiques. Le coût est principalement fonction de l'erreur qui est, pour un signal d'entrée donné, la différence entre la réponse désirée et la réponse réelle du réseau.

3.2.2 Le problème dû à l'inférence

Pour effectuer une détection de règle, les approches connexionnistes existantes ne semblent pas totalement adaptées : l'objectif visé dans les CMM [AUS88], [AUS89] est la reconnaissance de forme, le système doit être capable de faire de l'inférence, c'est-à-dire de tolérer une certaine différence entre la forme qui lui est présentée et la forme

[2] Un réseau peut comporter des milliers de neurones

qu'il a mémorisée. Dans son livre « Sparse Distributed Memory »[KAN88], Kanerva présente les SDM comme un système performant pour résoudre le problème du meilleur représentant. Or ici, on cherche à ne détecter que les données effectivement enregistrées. Cette tâche est difficile à implanter avec des réseaux de neurones et pour y parvenir, il est nécessaire d'adapter les modèles existants.

3.2.3 Adaptation à la détection de règles

Il existe deux problèmes lorsque l'on songe à réaliser un réseau de neurones permettant une identification formelle des données présentées :

- L'inférence qui est un inconvénient dans le cas de la détection des règles. Les réseaux de neurones sont conçus pour être tolérants au bruit ou pour être capables d'inférer une réponse : il faut donc les transformer, si possible, en système ne commettant aucune erreur. Pour s'assurer que les réseaux de neurones envisagés aient cette capacité, une étape de validation est requise.
- La taille de l'espace des entrées est immense et pose problème pour la validation. En effet, l'objectif est de manipuler des vecteurs de grande dimension : 128 bits voire même davantage. Or si nous calculons le nombre de vecteurs de 128 bits possibles, on en dénombre $2^{128}=3,4*10^{38}$: il est donc impossible d'effectuer une validation exhaustive dans un tel espace.

La première partie de la recherche consiste à tester des architectures avec des vecteurs de taille modeste (jusqu'à 16 bits) afin de pouvoir procéder à des tests exhaustifs : les tests mettent en évidence la présence d'erreurs.
Dans la deuxième partie, les réseaux de neurones ont été utilisés pour un prétraitement de données enregistrées dans une RAM : le réseau de neurones permet de déterminer à quelles adresses dans la RAM il est possible de trouver le vecteur présenté en entrée. Ce travail a été suggéré par le modèle développé par Zhou [ZHO99] qui applique

l'algorithme k-NN (section 3.3.1) dans un sous-espace sélectionné grâce à une CMM comme il est détaillé dans la section 4.5.1.

3.3 LE MODÈLE DES K-NN

Le modèle des k plus proches voisins ou « k-Nearest Neighbours » (k-NN) permet de classifier des données. En fonction des données apprises, le système décide à quelle classe appartient un vecteur d'entrée inconnu. Dans ce modèle, on étudie la distribution des classes au niveau local pour déterminer la classe du vecteur inconnu.

3.3.1 L'algorithme k-NN

Si l'on interprète l'ensemble des entrées possibles comme un espace E, un vecteur d'entrée $\vec{x}$ a une certaine classe d'appartenance. Dans le cas d'une tâche de classification avec l'algorithme k-NN, un enregistrement consiste en un couple ($\vec{v}$, $C(\vec{v})$) où $C(\vec{v})$ note la classe d'appartenance de $\vec{v}$.
Le principe de l'algorithme k-NN est d'identifier les k plus proches voisins[3] connus d'un point inconnu. L'algorithme attribue à ce point la classe trouvée en majorité parmi ces k plus proches voisins.

La Figure 3.2 présente l'exemple d'un algorithme 7-NN : ici, la classe de $\vec{v}$ peut être *A*, *B* ou *C*. Chaque cercle représente un vecteur de l'espace des entrées. Les cercles blancs représentent les vecteurs enregistrés et la lettre la classe à laquelle ils appartiennent. Le cercle gris est le point dont on cherche à déterminer la classe d'appartenance. Cette classe d'appartenance étant a priori inconnue, on la note X. Avec l'algorithme k-NN, on trouve que la classe inconnue X est la classe B car 4 des 7 plus proches voisins appartiennent à la classe B.

---

[3] Relativement à une métrique définie préalablement qui peut être une distance de Hamming, une distance euclidienne ou une autre

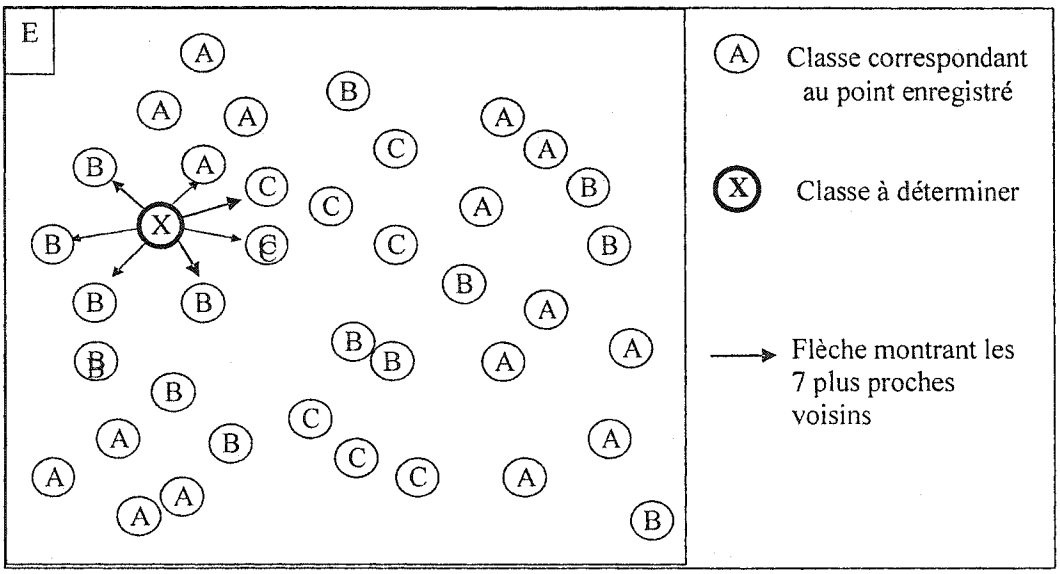


Figure 3.2 : Illustration de l'algorithme k-NN

3.3.2 Relation du contenu de la mémoire au vecteur d'entrée

On peut considérer, dans le modèle k-NN, que l'emplacement de stockage d'un élément est lié à son contenu : en effet, les vecteurs non enregistrés sont, d'une certaine manière stockés en mémoire par la connaissance de ses N plus proches voisins enregistrés. La sortie fournie par le système est donc dépendante de la place du vecteur d'entrée dans l'espace E. C'est pour cette raison que des vecteurs d'entrée proches ont une forte probabilité de donner des vecteurs de sortie proches, ce qui est, par exemple, totalement faux dans le cas d'une RAM.

3.3.3 L'interprétation des données

Le principal inconvénient de l'algorithme k-NN pour un problème tel que la détection de règles est qu'il interprète les données pour donner la sortie la plus cohérente possible relativement aux vecteurs connus : le système se trouve incapable de différencier les

informations inconnues des informations connues. De plus, cet algorithme demande un gros temps de traitement puisqu'il doit calculer la distance entre le vecteur d'entrée et chacun des vecteurs enregistrés.

3.4 LE MODÈLE DES CMM

Les CMM constituent un modèle de mémoires distribuées dans lequel les processus d'apprentissage et de récupération des enregistrements sont rapides. De plus, sous certaines conditions, ce modèle permet de retrouver exactement les données enregistrées. C'est ce modèle neuronal qui est utilisé dans tous les modèles de CAM neuronales étudiés par la suite. Pour comprendre quelles contraintes doivent être respectées, il est utile de connaître la théorie des CMM. Les coefficients de la matrice de corrélation peuvent être des réels, des entiers voire même de simples bits. Cette section est fortement inspirée du livre « Neural Networks: a Comprehensive Foundation» de Simon Haykin [HAY98].

3.4.1 Construction des CMM

La CMM peut se construire de manière itérative ou par un produit matriciel. Bien qu'elle ne soit pas la méthode utilisée, la méthode itérative permet de mieux comprendre le processus d'enregistrement et sera présentée en premier. Cette description permettra de comprendre aisément la construction par produit matriciel. Ensuite seront décrits le processus de récupération de l'information et les contraintes à respecter pour récupérer de manière exacte les données enregistrées.

3.4.1.1 Construction de la CMM par méthode itérative

Le nom de « matrice de corrélation » a pour origine la manière avec laquelle est construite cette matrice. Si l'on désire obtenir le vecteur $\vec{y}$ de sortie lorsque l'entrée

reçoit le vecteur $\vec{x}$, la CMM enregistre cette information en établissant une corrélation entre $\vec{x}$ et $\vec{y}$.

Supposons que l'on cherche à stocker n vecteurs $\{\vec{y}_1, \vec{y}_2, \ldots, \vec{y}_n\}$ aux adresses correspondantes $\{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\}$, alors la matrice de corrélation M vaut :

$$M = \sum_{k=1}^{n} \vec{y}_k \vec{x}_k^T \tag{3.4}$$

On constate ainsi que cette matrice est la superposition du produit externe de chacun des couples $(\vec{x}_k, \vec{y}_k)$ et que l'ordre d'enregistrement des différentes données n'influence pas la valeur des coefficients de la matrice. On peut également voir que les CMM sont des mémoires distribuées : l'enregistrement de chaque donnée est réparti sur toute la matrice.

3.4.1.2 Construction de la CMM par produit matriciel

L'expression matricielle pour la construction de la CMM est, en fait, un « batch » de la méthode itérative. Soient les matrices X et Y telles que:

$$X = [\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n] \tag{3.5}$$

$$Y = [\vec{y}_1, \vec{y}_2, \ldots, \vec{y}_n] \tag{3.6}$$

On peut alors exprimer M comme un produit matriciel :

$$M = YX^T \tag{3.7}$$

Pour retrouver les sorties correctement à la lecture, il est important de normaliser les vecteurs d'entrée $\vec{x}_k$

3.4.2 Récupération des enregistrements dans la CMM

Le processus de lecture dans une CMM est simple, il consiste en un simple produit matriciel. Ainsi, la sortie $\vec{y}$ correspondant à un vecteur d'entrée $\vec{x}$ sera :

$$\vec{y} = M\vec{x} \tag{3.8}$$

Sous certaines conditions, il est possible de retrouver un vecteur $\vec{y}_j$ préalablement enregistré si l'on présente le vecteur $\vec{x}_j$ lui correspondant en entrée. Si on développe l'équation 3.4 pour le vecteur $\vec{x}_j$ on obtient :

$$\begin{aligned} \hat{\vec{y}}_j &= \sum_{i=1}^{n} \vec{y}_i \vec{x}_i^T \vec{x}_j \\ &= \sum_{i=1}^{n} \vec{y}_i \left(\vec{x}_i^T \vec{x}_j \right) \\ &= \sum_{i=1}^{n} \left(\vec{x}_i^T \vec{x}_j \right) \vec{y}_i \end{aligned} \tag{3.9}$$

On peut commuter le terme $\left(\vec{x}_i^T \vec{x}_j \right)$ qui est un produit scalaire.

Ensuite, on peut écrire :

$$\hat{\vec{y}}_j = \left(\vec{x}_j^T \vec{x}_j \right) \vec{y}_j + \sum_{\substack{i=1 \\ i \neq j}}^{n} \left(\vec{x}_i^T \vec{x}_j \right) \vec{y}_i \tag{3.10}$$

Si on a normalisé les vecteurs à l'enregistrement, on a les relations suivantes :

$$\vec{x}_j^T \vec{x}_j = \left\| \vec{x}_j \right\|^2 = 1 \tag{3.11}$$

$$\vec{x}_i^T \vec{x}_j = \cos(\vec{x}_i, \vec{x}_j) \tag{3.12}$$

Ainsi, l'équation (3.10) devient :

$$\hat{\vec{y}}_j = \vec{y}_j + \sum_{\substack{i=1 \\ i \neq j}}^{n} \cos\left(\vec{x}_i, \vec{x}_j \right) \vec{y}_i \tag{3.13}$$

3.4.3 Représentation neuronale d'une CMM

Une CMM peut être représentée en tant que réseau de neurones (Figure 3.3) : les poids synaptiques sont ajustés avec $\Delta\omega_{ij} = x_j y_i$ pour chaque échantillon appris. La

représentation neuronale d'une CMM montre comment la mémoire est distribuée sur chacun des neurones.

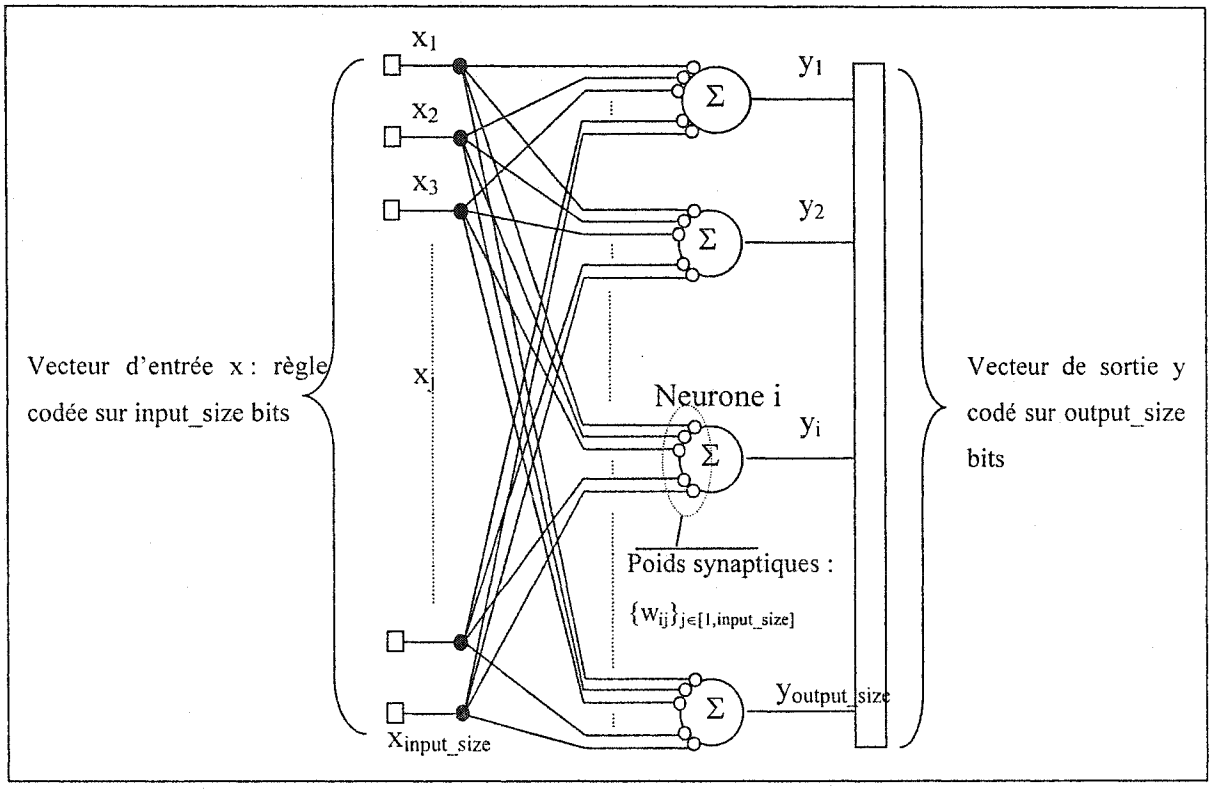


Figure 3.3 : Représentation neuronale d'une CMM

3.4.4 Minimisation de la corrélation croisée

Comme le montre l'équation (3.13), on est assuré de retrouver $\vec{y}_j$ si les vecteurs d'entrée sont orthonormaux deux à deux. Toutefois, $\hat{\vec{y}}_j$ sera proche de $\vec{y}_j$ si $\left\| \sum_{\substack{i=1 \\ i \neq j}}^{n} \cos(\vec{x}_i, \vec{x}_j)\vec{y}_i \right\|_\infty \leq \left\| \vec{y}_j \right\|_\infty$. Par la suite, ce cas de figure sera évoqué sous le nom de pseudo orthogonalité.

Ainsi, deux approches seront étudiées :

- L'orthogonalisation des vecteurs d'entrée
- La transformation des vecteurs d'entrée en vecteurs « sparse » et distribués uniformément sur l'espace, caractéristiques qui assurent une pseudo orthogonalité.

3.5 CONCLUSION

Les réseaux de neurones, en raison de leurs propriétés spécifiques, offrent à la fois des avantages et des inconvénients dans la perspective de leur utilisation pour des systèmes de détection de règles. Leur capacité à inférer une réponse en présence de vecteurs d'entrée inconnus, qui est un avantage dans leurs utilisations habituelles, est dans ce cas un inconvénient majeur. En revanche, les méthodes de stockage de l'information employées dans les réseaux de neurones permettent de relier fortement les données à leur lieu de stockage. De plus, des modèles de mémoire tel celui des CMM permet de stocker l'information des vecteurs enregistrés de manière globale : ainsi l'étape de prétraitement est globale au lieu d'être spécifique à chaque cellule d'enregistrement comme dans les CAM « traditionnelles ».

CHAPITRE 4

ADAPTATION DES MODÈLES DE CMM EXISTANTS

4.1 INTRODUCTION

Dans le cadre de cette recherche, cinq modèles utilisant des CMM ont été explorés. Le chapitre présente les différentes simulations effectuées et détermine les configurations valides et fournit un certain nombre de caractéristiques qui permettent de donner une estimation de coût et de performances de ces différents modèles. Les trois premiers modèles utilisent un procédé d'orthogonalisation exact, les deux derniers, un procédé de projection des vecteurs dans un espace de grande dimension.

L'étude des différents modèles suit la démarche adoptée pour aboutir à des systèmes rapides et peu exigeants en ressources. Cette démarche a consisté en des raffinements successifs du modèle d'origine. En utilisant les informations fournies par les modèles étudiés antérieurement, on a pu dégager des contraintes supplémentaires à respecter et ainsi adapter le modèle à ces contraintes.

Les simulations décrites dans ce chapitre sont constituées de deux phases : la première consiste en une validation du modèle proposé, la seconde a pour but de fournir les informations nécessaires au dimensionnement de la CMM. Ce dimensionnement permet de déterminer les ressources mémoires nécessaires pour stocker les coefficients de la CMM et le nombre d'opérations requises pour effectuer une lecture dans la CMM. Il faut noter qu'aucune estimation du coût de mise en mémoire des données n'est faite car on cherche à obtenir un système très rapide en lecture seulement. La mise en mémoire

des données n'a pas de contraintes de rapidité et ne nécessite que des ressources logicielles (calcul des coefficients de la CMM).

4.2 ORTHOGONALISATION DES VECTEURS COMPLÉMENTÉS

Comme il a été vu dans la section 3.4.2, si les vecteurs enregistrés dans la CMM sont orthogonaux deux à deux, on est assuré de retrouver une réponse exacte pour les vecteurs enregistrés. La première idée est donc d'orthogonaliser la famille de vecteurs à enregistrer.

4.2.1 Le principe de la décomposition QR

4.2.1.1 Procédé de Gram-Schmidt et méthode QR

Le procédé d'orthogonalisation de Gram-Schmidt permet de transformer une famille de vecteurs linéairement indépendants en une famille orthonormale de vecteurs.

Soit $\{\vec{e}_i\}_{i\in[1,d]}$ une famille libre de vecteurs, alors la famille $\{\vec{\varepsilon}_i\}_{i\in[1,d]}$ définie par récurrence suivante est orthonormale :

$$\begin{cases} \vec{\varepsilon}_1 = \dfrac{\vec{e}_1}{\|\vec{e}_1\|} \\ \forall k \in [1, d-1] \;\; \vec{\varepsilon}_{k+1} = \dfrac{\vec{e}_{k+1} - \sum_{i=1}^{k} \langle \vec{e}_{k+1}, \vec{\varepsilon}_i \rangle \vec{\varepsilon}_i}{\left\| \vec{e}_{k+1} - \sum_{i=1}^{k} \langle \vec{e}_{k+1}, \vec{\varepsilon}_i \rangle \vec{\varepsilon}_i \right\|} \end{cases}$$

Avec $\langle \vec{x}, \vec{y} \rangle$ notant le produit scalaire de $\vec{x}$ par $\vec{y}$.

Le procédé de Gram-Schmidt montre que tout vecteur $\vec{e}_k$ peut être exprimé en fonction des vecteurs $\{\vec{\varepsilon}_1, \ldots, \vec{\varepsilon}_k\}$.

Notons, dans le procédé de Gram-Schmidt, n la dimension des vecteurs $\vec{e}_k$ et $\vec{\varepsilon}_k$. Notons M la matrice nxd telle que $M = (\vec{e}_1, \ldots, \vec{e}_d)$ et Q la matrice orthonormale $Q = (\vec{\varepsilon}_1, \ldots, \vec{\varepsilon}_d)$. Compte tenu de la remarque précédente, il existe une matrice triangulaire supérieure R de taille dxd telle que $M = QR$. C'est cette décomposition de M en un produit d'une matrice orthonormale Q et d'une matrice triangulaire supérieure R par le procédé de Gram-Schmidt que l'on nomme décomposition QR.

La décomposition QR utilisée dans les simulations est une fonction standard fournie par le logiciel Matlab. Une description précise de l'algorithme utilisé pour obtenir la décomposition QR se trouve dans le livre « Numerical Recipes in C » [PRE93].

4.2.1.2 Application à l'orthogonalisation des règles

Dans le cas de règles, on désigne la transposée de la matrice des entrées X (section 3.4.1) par IN et la transposée de la matrice des sorties Y par OUT. Il est nécessaire d'utiliser la transposée afin de pouvoir utiliser correctement la décomposition QR.

En effet, soient {IN,OUT} un ensemble de règles telles que :

$$IN = \begin{bmatrix} \text{Règle}_1 \\ \text{Règle}_2 \\ \vdots \\ \text{Règle}_n \end{bmatrix} \text{ et } OUT = \begin{bmatrix} \text{Sortie}_1 \\ \text{Sortie}_2 \\ \vdots \\ \text{Sortie}_n \end{bmatrix}$$

Une règle est codée sur *input_size* bits et une sortie sur *output_size* bits. Il faut noter que les règles sont décrites par des vecteurs lignes : il faudra donc transposer IN et OUT pour utiliser la théorie sur les CMM.
Comme nous l'avons vu dans la section 3.4.4, pour retrouver une sortie absolument non bruitée avec une CMM, il faut que la matrice des entrées à enregistrer soit orthonormale : on va pour cela utiliser la décomposition QR décrite précédemment.

On va donc chercher à enregistrer Q et non IN, soit $Q=IN.R^{-1}$ dans la CMM. Or, pour pouvoir obtenir une matrice R inversible[4], on doit transformer IN et OUT légèrement. En effet, il est nécessaire que la matrice IN soit carrée et qu'elle soit constituée de vecteurs libres : il s'agit donc de compléter l'ensemble des règles avec une famille linéairement indépendante de pseudo-règles dont aucune n'est une combinaison linéaire des règles existantes.

$$IN' = \left[\begin{array}{c} \text{Règle}_1 \\ \vdots \\ \text{Règle}_n \\ \hline \text{Règle}_{n+1} = \text{ Pseudo Règle} \\ \vdots \\ \text{Règle}_{input\_size} = \text{ Pseudo Règle} \end{array}\right] \text{ et } OUT' = \left[\begin{array}{c} \text{Sortie}_1 \\ \vdots \\ \text{Sortie}_n \\ \hline \text{Sortie nulle} \\ \vdots \\ \text{Sortie nulle} \end{array}\right]$$

Il faut remarquer qu'il est nécessaire de pouvoir définir la $i^{ème}$ règle en fonction du $i^{ème}$ vecteur ligne de Q et de la matrice R. En effet, si la création de la CMM se fait en connaissant tous les vecteurs à enregistrer, le processus de récupération se fait pour un vecteur seulement : ce vecteur, une fois transformé doit correspondre à un des vecteurs effectivement enregistré dans la CMM. Si l'on ne tient pas compte de ce problème, on risque d'obtenir dans Q des combinaisons linéaires des vecteurs enregistrés.

Notons respectivement in'_{ij}, q_{ij} et r_{ij} les coefficients situés à $i^{ème}$ ligne et $j^{ème}$ colonne de IN', Q, R. Le produit matriciel donne :

$$\forall i, j \in [1, input\_size]\ in'_{ij} = \sum_{k=1}^{input\_size} q_{ik} r_{kj} \qquad (4.1)$$

Si les règles sont décrites par des vecteurs colonne de IN', chaque règle de IN' dépend de tous les coefficients de Q. En revanche, si les règles sont décrites par des vecteurs

---

[4] Si on a une matrice carrée constituée de vecteurs libres, on obtiendra une matrice R inversible [PRE93]

lignes de *IN'*, la i^{ème} règle est globalement fonction du i^{ème} vecteur ligne de *Q* et de *R* : le i^{ème} vecteur ligne de Q est donc équivalent à la i^{ème} règle. La Figure 4.1 illustre ce processus.

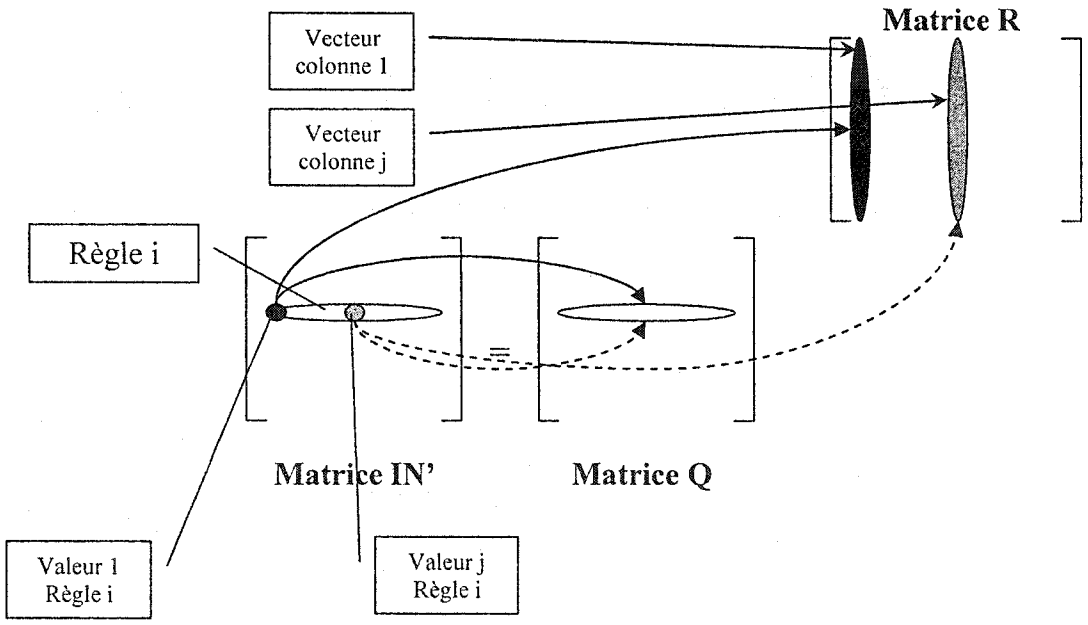


Figure 4.1 : Correspondance entre la matrice IN' et la matrice Q

Dans la théorie des CMM décrite dans la section 3.4, on utilise des vecteurs colonne soit $X=Q^T$.

En aménageant l'expression de l'équation à notre problème, nous obtenons :

$$M=OUT'^T.Q=OUT'^T.IN'.R^{-1} \tag{4.2}$$

Pour retrouver le vecteur, on effectuera pour un vecteur colonne $\vec{x}$ en entrée :

$$\hat{Y} = M.(\vec{x}^T.R^{-1})^T = M.(R^{-1})^T\vec{x} = M'\vec{x} \tag{4.3}$$

avec

$$M' = M.(R^{-1})^T \tag{4.4}$$

On constate qu'une fois M' calculée, il suffit de tester un vecteur sans le transformer car le changement de base est compris dans le calcul de M'.

4.2.1.3 La transformation des entrées utilisées

Lorsqu'on crée les règles, il est fréquent que les vecteurs correspondants constituent une famille liée. Dans ce cas, le rang de la matrice R sera dégénéré et R sera, par conséquent, non inversible. Pour éviter ce problème, il est nécessaire de transformer les vecteurs d'entrée en une famille de vecteurs linéairement indépendants. Si le processus d'orthogonalisation se fait avec des nombres réels, les entrées du système sont binaires.

Ainsi, la première approche envisagée consiste à transformer le vecteur d'entrée binaire $\vec{X}$ en concaténant l'entrée d'origine avec son complément $\overline{\vec{X}}$ (Figure 4.2). Cette transformation est envisagée car elle permet de créer une famille de vecteurs linéairement indépendants.

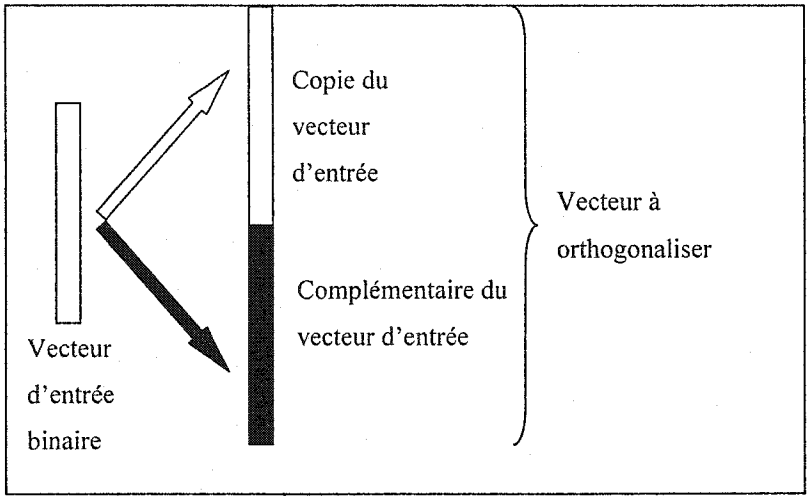


Figure 4.2 : Transformation d'un vecteur par adjonction du complémentaire

Si, par exemple, on enregistre les vecteurs A=(1,0,0,0) et B=(0,0,1,0). On obtient, après transformation, les vecteurs A'=(1,0,0,0,0,1,1,1) et B'=(0,0,1,0,1,1,0,1). Si l'on considère le vecteur (1,0,10) qui est une combinaison linéaire de A et de B, on constate que le vecteur transformé correspondant, (1,0,1,0,0,1,0,1) n'est pas une combinaison linéaire de A' et B'.

Cette transformation est adoptée car, si elle double la taille des vecteurs, elle est très facile à implanter tant au niveau logiciel que matériel car fait appel à des transformations élémentaires. C'est à partir de ces vecteurs transformés que l'on procède à l'orthogonalisation.

4.2.2 Processus de validation

Les simulations ont pour objectif de déterminer si le modèle décrit ci-dessus est valide. Pour cela, on va comparer la réponse donnée par la CMM avec la réponse qu'elle est supposée donner : la méthode de comparaison employée est détaillée dans la section 4.2.2.2. En outre, on analysera les erreurs notamment en différenciant celles de type I de celles de type II.

4.2.2.1 Les vecteurs enregistrés dans la CMM

Les vecteurs d'entrée de la CMM sont les vecteurs transformés par adjonction du complémentaire, les vecteurs de sortie sont pris identiques aux entrées, ce sont donc aussi les vecteurs transformés : on parle de CMM auto associative.

4.2.2.2 Principe de la méthode de validation

Afin de vérifier que la réponse du modèle simulé est correcte, un système non neuronal détermine la réponse attendue. On fera référence à ce système par le terme de

« vérificateur ». Pour déterminer si la CAM neuronale donne une réponse correcte, il suffit de comparer sa sortie à celle donnée par le système vérificateur (Figure 4.3).

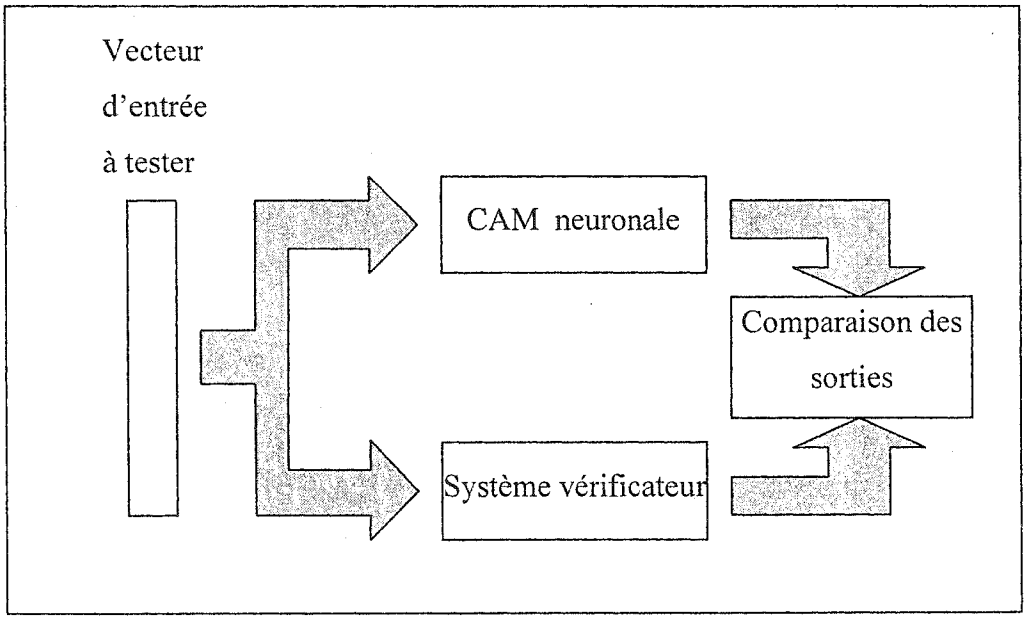


Figure 4.3 : La méthode de vérification des vecteurs de sortie

Ainsi, ce dispositif permet de déterminer si la CAM neuronale commet ou non des erreurs. Ces erreurs peuvent être de différentes natures et il faut collecter le maximum d'informations pouvant servir à comprendre les causes d'erreur.

Les différents types d'erreurs. Pour pouvoir plus facilement interpréter les causes d'erreurs, il est important de distinguer plusieurs types d'erreurs : on peut différencier les erreurs produites avec des vecteurs enregistrés (erreurs de type I) des erreurs produites avec des vecteurs non enregistrés (erreurs de type II). Une différenciation plus précise peut être réalisée en classant les erreurs selon la distance de Hamming à l'enregistrement le plus proche. Cette classification permet d'identifier si les erreurs se

produisent avec des vecteurs enregistrés et permet en plus d'évaluer l'influence de la distance de Hamming sur l'erreur.

La notion de tolérance. Etant donné le choix de la sortie, l'appariement se réduit à comparer la sortie obtenue à l'entrée présentée au système. Compte tenu des erreurs d'arrondi, on tolère une différence δ[5] entre l'entrée présentée $\vec{x}$ et la sortie obtenue $\vec{y}$ et on considère qu'il y a appariement si

$$\|\vec{x} - \vec{y}\|_{\infty} < \delta \tag{4.5}$$

C'est-à-dire si chacune des composantes du vecteur différence $\vec{x}$ - $\vec{y}$ est inférieure à δ (Figure 4.4). Le δ est fixé a posteriori afin que tous les vecteurs enregistrés soient détectés correctement.

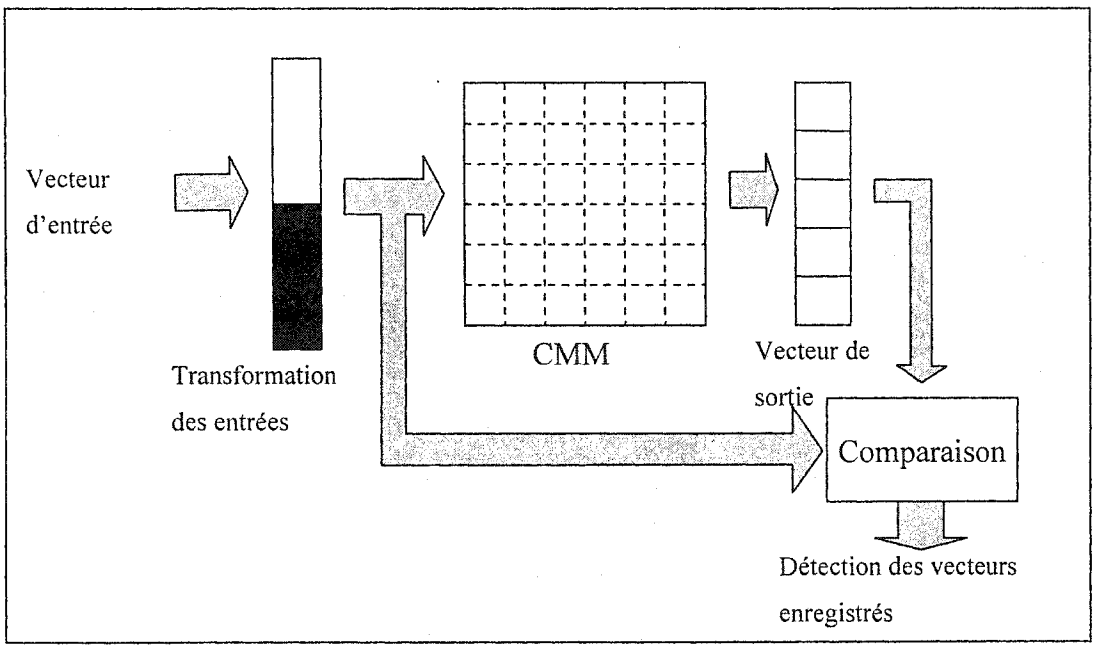


Figure 4.4 : Modèle employé avec la méthode de complémentation des entrées

[5] Ce δ sera parfois désigné par le terme « tolérance »

4.2.2.3 Paramètres de la simulation

Le problème de la validation : validation exhaustive ou non exhaustive. Pour valider un modèle, il faut tester le comportement du système dans un certain nombre de configurations types. Dans le cadre des simulations effectuées, il existe deux alternatives pour valider les architectures. La première est la validation exhaustive qui consiste à tester toutes les entrées possibles avec tous les enregistrements possibles : en cas de succès avec cette méthode, on est assuré que le système répondra correctement dans tous les cas de figure possibles. Malheureusement, même pour des tailles de vecteurs d'entrée modestes, une validation exhaustive est irréalisable (Annexe III). Dans le cas d'espaces trop grands, il est seulement possible de réaliser une validation partielle qui donnera une estimation de la fiabilité du système mais ne pourra pas garantir le bon comportement du système pour tous les cas de figure possibles.

Choix d'une validation exhaustive. Il a été choisi de procéder à une validation exhaustive avec de petits vecteurs d'entrée afin de pouvoir étudier plus facilement les erreurs éventuellement commises par le système : la simulation présentée a été effectuée avec des vecteurs d'entrée de 4 bits : tous les enregistrements possibles ont été testés pour chacun des états possibles de la CMM –on peut enregistrer entre un et quatre vecteurs- soit 40256 tests (Annexe III).

4.2.3 Résultats : le problème des combinaisons linéaires

Si le système se comporte parfaitement en présence des entrées enregistrées, il commet des erreurs avec des entrées non enregistrées et la fréquence de ces erreurs est d'autant plus élevée que le nombre de vecteurs enregistrés est grand (Tableau 4.1). Une étude des cas d'erreurs nous montre que celles-ci sont systématiquement causées par des combinaisons linéaires d'entrées enregistrées (Tableau 4.2). En outre, lorsque ces erreurs

se produisent, la différence entre l'entrée présentée et la sortie non arrondie est de l'ordre de 10^{-1}, ce qui est une différence difficilement détectable.

Tableau 4.1 : Pourcentage d'erreurs en fonction du nombre d'enregistrements

Nombre d'enregistrements	Nombre d'états	Pourcentage d'erreurs
1	16	0
2	120	0,05
3	560	4,35
4	1820	18,5

Dans les exemples du Tableau 4.2, les combinaisons linéaires sont faciles à déterminer pour les 4 premiers cas. Dans le cas n°5, les combinaisons linéaires sont un petit peu plus difficiles à trouver et on a :

(0,1,0,0) = (1,1,0,0)-(1,0,0,0)

(0,0,1,0) = (1,0,0,0)+(0,1,1,0)-(1,1,0,0)

(1,0,1,0) = 2*(1,0,0,0)+(0,1,1,0)-(1,1,0,0)

(1,1,1,0) = (1,0,0,0)+(0,1,1,0)

Tableau 4.2 : Exemple d'erreurs détectées avec des vecteurs de 4 bits complémentés

	Cas n°1	Cas n°2	Cas n°3	Cas n°4	Cas n°5
Vecteurs enregistrés avant transformation	0 1 1 0 1 1 1 1	0 0 0 0 1 0 0 0 0 1 0 0	0 0 0 0 1 0 0 0 1 1 0 0	0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0	0 0 0 0 1 0 0 0 1 1 0 0 0 1 1 0
Vecteurs provoquant des erreurs	1 0 0 1	1 1 0 0	0 1 0 0	1 1 0 0 1 0 1 0 0 1 1 0 1 1 1 0	0 1 0 0 0 0 1 0 1 0 1 0 1 1 1 0

4.2.4 Une mise en évidence du problème de la corrélation croisée

Il faut retenir de cette série de simulations sur les CMM que si l'orthogonalisation des vecteurs d'entrée enregistrés permet de retrouver les sorties qui leur correspondent, le résultat obtenu avec les vecteurs non enregistrés n'est pas prévisible et n'est pas toujours celui désiré : la CMM tente de construire la réponse que devrait donner le système et parvient à trouver la réponse que devrait donner le système dans ce cas particulier, en donnant en sortie le vecteur d'entrée et son complémentaire pour certains vecteurs inconnus. Ce problème est connu sous le nom de corrélation croisée, ce que Cooper appelle la logique animale [COO73]. Etant donné que l'on cherche un système ne commettant aucune erreur, on ne peut accepter ce modèle pour retrouver des règles enregistrées.

Constatant l'inadaptation de ce modèle à notre problème, une autre méthode a été explorée pour laquelle, on cherche à coder de manière différente les différents bits d'un vecteur.

4.3 ORTHOGONALISATION PAR RECODAGE CARACTÉRISTIQUE DE LA COMPOSANTE

Ainsi, dans ce deuxième modèle, si on utilise encore la décomposition QR (section 4.2.1), on modifie la méthode de transformation des vecteurs d'entrée. Dans le modèle utilisant l'adjonction du complémentaire (section 4.2), le système commet, accidentellement, des erreurs de type II (section 4.2.3) : par construction de la CMM, chaque bit d'un vecteur d'entrée enregistré influence tous les bits obtenus en sortie. Pour pallier aux erreurs du modèle utilisant l'adjonction du complémentaire, on a intuitivement cherché à recoder les vecteurs d'entrée afin de caractériser les bits tant par leur valeur ('0' ou '1') que par la composante qu'ils représentent.

4.3.1 Principe du recodage

Modification du modèle. La transformation employée est un recodage des entrées qui permet de coder de manière unique chaque '1' du vecteur d'origine[6] : en fait, le '1' est codé linéairement en fonction de l'emplacement du bit dans le vecteur. Si cette transformation n'est pas la seule possible, c'est l'une des premières transformations qui a été envisagée et qui permet d'obtenir un modèle valide. Le '0' est, lui, toujours codé par la valeur 1 : on peut considérer qu'une valeur de 0 marque une absence d'information, ce qui n'est pas représentatif de l'information qu'apportent effectivement les '0' du vecteur d'entrée. De plus, la transformation utilisée a l'avantage de garder la même taille de vecteur d'entrée et donc de faire des économies au niveau de la taille de la CMM.

---

[6] Il serait envisageable de coder tous les zéros différemment, mais cela ne semble pas nécessaire

Notons $\{\vec{b}_i\}_{i\in[1,n]}$ la base canonique de l'espace des entrées $\{0,1\}^n$ et soit $\vec{x} = \sum_{k=1}^{n} x_k \vec{b}_k \in \{0,1\}^n$ un vecteur d'entrée du système. Le vecteur binaire $\vec{x}$ est transformé en un vecteur à coefficients entiers $\vec{x}'$ de la manière suivante :

$$\vec{x}' = \sum_{k=1}^{n} (1 + k.x_k)\vec{b}_k \tag{4.6}$$

Il faut noter que la capacité maximale de la CMM est alors, pour une entrée de n bits, n-1. En effet, il faudrait que le $n^{ième}$ vecteur complète la base de l'espace, ce qui ne peut être le cas pour toutes les entrées possibles.

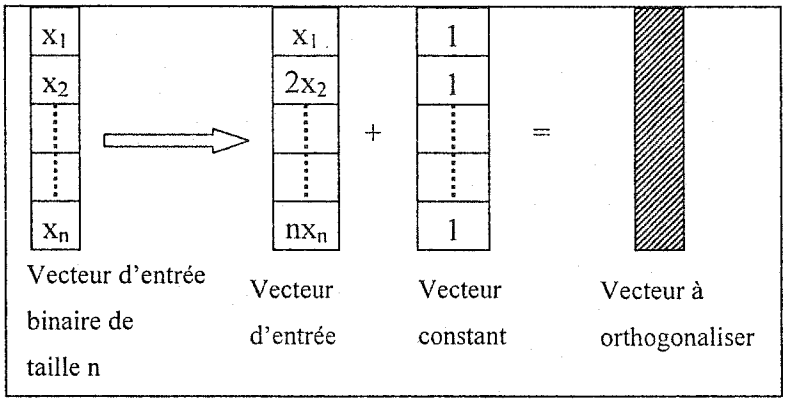


Figure 4.5 : Transformation d'un vecteur par recodage caractéristique de la composante

La suppression des cas de combinaisons linéaires. Supposons que l'on ait enregistré les vecteurs (1,0,0,0) et (0,1,0,0), cas qui a provoqué une erreur dans la simulation précédente (Tableau 4.2) lorsque le vecteur (1,1,0,0) a été présenté. Les vecteurs, une fois transformés, vont valoir respectivement (2,1,1,1) et (1,3,1,1). Le vecteur (1,1,0,0) devient (2,3,1,1). Après transformation, ce vecteur n'est plus une combinaison linéaire des 2 vecteurs précédents.

4.3.2 Méthode de validation et de détermination des ressources mémoires nécessaires

La première partie de cette section décrit la méthode utilisée pour valider le modèle utilisant un recodage caractéristique de la composante. La deuxième partie montre l'influence de la précision des coefficients sur les ressources mémoires à allouer aux coefficients de la CMM et explique la relation entre la tolérance δ (section 4.2.2.2) et les ressources mémoires nécessaires.

4.3.2.1 Les vecteurs enregistrés dans la CMM

Le schéma de fonctionnement de la CAM neuronale est identique à celui adopté dans la section 4.2.2. La sortie enregistrée est identique à l'entrée : *OUT=IN* et la phase d'appariement compare également l'entrée et la sortie en acceptant une certaine tolérance δ entre la réponse obtenue et l'entrée.

4.3.2.2 Les tests effectués

Des tests exhaustifs ont été effectués avec des vecteurs d'entrée de 4 bits. Ensuite, 10 essais ont été effectués pour chacune des tailles de vecteurs d'entrée 8, 16, 32, 64 et 128 bits. Dans ces cas de figure, il est impossible de procéder à une validation exhaustive (Annexe III). Ainsi, seule une validation partielle a été effectuée : pour une taille d'entrée de n bits, on a effectué une simulation avec 1 vecteur enregistré, 2 vecteurs enregistrés,... jusqu'à (n-1) vecteurs enregistrés. Les vecteurs utilisés dans la phase de validation étaient :

- Tous les vecteurs enregistrés
- Tous les vecteurs situés à une distance de Hamming de 1 d'un des vecteurs enregistrés soit les plus susceptibles de provoquer des erreurs.

4.3.2.3 Coût des coefficients de la CMM

Le coût en ressources mémoire à allouer à la CMM est fortement dépendant de la précision nécessaire afin de ne pas commettre d'erreurs. Il faut, en fait, réussir à trouver la tolérance δ (section 4.2.2.2) optimale qui permette de ne pas commettre d'erreurs d'une part et de minimiser les ressources mémoires à allouer aux coefficients de la CMM d'autre part. En effet, plus la précision nécessaire est petite, plus le besoin en ressources mémoire à allouer aux coefficients de la CMM est bas.

Supposons que la CMM soit constituée de coefficients à virgule fixe. Soit c la valeur maximale parmi les coefficients de la CMM et d le nombre de décimales des coefficients, alors le nombre de bits nécessaires pour coder c est *nb_bits* défini par :

$$nb\_bits = \left\lceil \log_2 \left(c * 10^d \right) \right\rceil \tag{4.7}$$

où $\lceil \bullet \rceil$ note la fonction de plafonnage qui arrondit à l'entier supérieur. On constate que la taille mémoire à allouer à chaque coefficient est fonction de la valeur maximale trouvée parmi les coefficients de la CMM d'une part et de la précision des coefficients d'autre part.

4.3.3 Un système sans erreurs apparentes

Afin que les ressources mémoires à allouer à la CMM soient les moins grandes possibles, il faut chercher la plus grande tolérance qui ne provoque pas d'erreurs. On peut noter que la tolérance maximale est $\delta = 0{,}5$, ce qui équivaut à arrondir les sorties obtenues.

Essai exhaustif à 4 bits. Des tests exhaustifs ont été réalisés avec 4 bits en entrée : toutes les entrées possibles ont été testées avec tous les enregistrements possibles. Aucune erreur n'a été détectée avec une tolérance $\delta = 0{,}1$ sur la sortie obtenue. Contrairement au

cas des entrées complémentées, l'utilisation du recodage caractéristique des composantes permet de ne plus avoir d'erreurs pour des vecteurs de 4 bits.

Essai à 8 bits. L'ensemble des vecteurs enregistrés contient de 1 à 7 éléments. Ensuite la validation se fait en testant les 256 vecteurs d'entrée possibles. Pour des vecteurs d'entrée de 8 bits, on constate que des erreurs se produisent avec une tolérance de $\delta = 0{,}1$. Toutefois, ces erreurs ne sont présentes que pour des vecteurs situés à une distance de un des vecteurs enregistrés. Le problème n'existe plus avec une tolérance $\delta = 0{,}01$. De manière plus générale, la fréquence d'erreur semble fonction de la distance de Hamming minimale entre le vecteur d'entrée et les différents vecteurs enregistrés : plus un vecteur est proche d'un vecteur enregistré, plus il semble susceptible de générer une erreur. Cette série d'essais montre qu'il est nécessaire d'avoir une tolérance $\delta = 0{,}01$ pour ne pas commettre d'erreurs. La contrainte sur la tolérance suggère ainsi 2 séries de simulations :

- La première consiste à étudier l'influence de la distance de Hamming du plus proche enregistrement sur la fréquence d'erreur.
- La seconde consiste à étudier la tolérance en fonction de la taille de l'entrée.

Influence de la distance de Hamming sur la fréquence d'erreurs. Afin de pouvoir apprécier l'influence de la distance de Hamming sur la fréquence des erreurs, une série de simulations a été effectuée avec une très grande tolérance (Figure 4.6) : le vecteur de sortie est transformé par un simple arrondi de ses coefficients, c'est-à-dire $\delta = 0{,}5$. Le choix de cette très grande tolérance a pour but d'accentuer le nombre d'erreurs et de mieux observer l'influence de la distance de Hamming. Des simulations avec des vecteurs d'entrée de 8, 10 et 12 bits ont été réalisées. Pour chaque expérience, 100 simulations par nombre d'enregistrements possibles ont été effectuées. Par exemple, pour une taille d'entrée de 8 bits, 100 essais ont été effectués avec 1 vecteur enregistré, 100 avec 2 vecteurs enregistrés jusqu'à 100 essais avec 7 vecteurs enregistrés. La validation a été exhaustive : tous les vecteurs possibles ont été testés au cours de la

validation pour chacun des tests. Les résultats obtenus mettent en évidence l'influence de la distance de Hamming au plus proche vecteur enregistré sur la fréquence d'erreur. Ainsi les erreurs les plus probables sont celles produites pour des vecteurs situés à une distance de Hamming de 1 du plus proche des vecteurs enregistrés. On constate que le système ne commet aucune erreur de type I (ce qui correspond à une distance de Hamming de zéro). Ceci s'explique grâce aux propriétés des CMM (section 3.4.4) qui garantissent de retrouver les bons vecteurs de sortie pour un ensemble de vecteurs d'entrée orthogonaux.

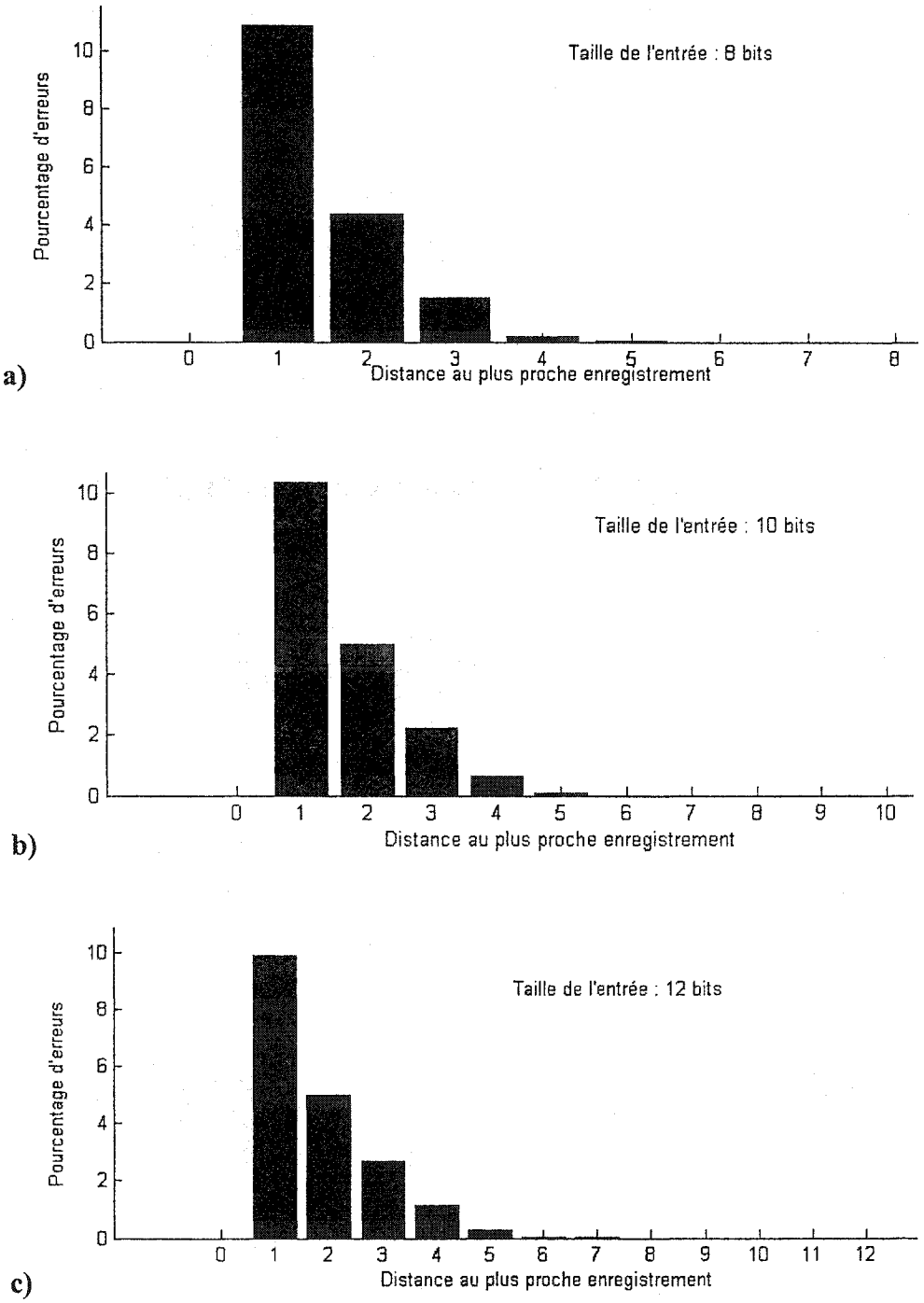


Figure 4.6 : Fréquence d'erreur en fonctions de la distance au plus proche enregistrement. a) Taille d'entrée de 8 bits. b) Taille d'entrée de 10 bits. c) Taille d'entrée de 12 bits.

Simulations en grande dimension. Dans toutes les simulations effectuées, on pouvait ne commettre aucune erreur à condition que la précision des calculs soit assez grande. On constate que le réglage de la tolérance est critique et il semble d'autant plus difficile de l'ajuster que la taille des vecteurs est grande. Toutefois le problème principal de la validation en grande dimension est que l'on ne peut pas vraiment garantir le bon fonctionnement du système.

4.3.4 Un système impossible à valider exhaustivement

Comme il a été mentionné précédemment, même si le système semble ne commettre aucune erreur, il n'est possible de garantir la réponse du système que dans le cas des vecteurs enregistrés. On peut vérifier les vecteurs situés à une distance de 1 au sens de Hamming, ce qui constitue déjà un grand nombre d'essais : si la taille de l'entrée est de 128 bits et que 127 vecteurs sont enregistrés, il peut y avoir jusqu'à 127*128 = 16256 vecteurs à tester. Mais il devient très difficile de tester tous les vecteurs situés à une distance de Hamming de 2 : il y a $127*\binom{128}{2} = 127*\frac{128!}{2!(128-2)!} = 1\,032\,256$ vecteurs de ce genre. Il est donc impossible de garantir le bon comportement du système pour de grandes dimensions, même si un système qui ne commet aucune erreur pour tous les vecteurs distants de 1 des vecteurs enregistrés a une très faible probabilité de commettre des erreurs dans d'autres cas.

4.4 Orthogonalisation d'une CMM pointant dans une RAM

Les problèmes de validation en grande dimension rencontrés à la section 4.3.4 suggèrent la recherche d'une méthode de validation qui permette de certifier le bon fonctionnement du système dans tous les cas de figure. Or la méthode de validation dépend directement de l'architecture envisagée. Une nouvelle architecture a donc été

développée : si l'on considère l'architecture précédente, on constate que l'appariement se fait entre le vecteur d'entrée et la réponse inférée par le système. Etant donné la complexité de la transformation des entrées et les caractéristiques de non linéarité des réseaux de neurones, il est impossible d'analyser le comportement du système pour tous les cas de figure possibles. Si dans cette simulation, le modèle utilisé pour la CMM est le même que celui décrit dans la section 4.3, la construction de la CMM est différente puisque la sortie fournit une adresse qui correspond à un emplacement mémoire dans une RAM. Cette architecture permet de certifier le bon fonctionnement du système a posteriori (section 4.4.1.1).

4.4.1 Fonctionnement de l'architecture

4.4.1.1 Principe de détection des règles

L'architecture est modifiée de telle sorte que la CMM exécute seulement le prétraitement qui fournit l'adresse possible d'enregistrement de la donnée. L'appariement, qui se situe au niveau de la RAM, consiste à comparer le vecteur d'entrée non transformé et le vecteur de la RAM sélectionné par la CMM (Figure 4.7).

Cette modification permet d'avoir une étape de validation beaucoup plus simple où il suffit de vérifier que le vecteur d'entrée non transformé est égal au vecteur de la RAM pointé par la CMM. En effet, lorsqu'un vecteur non enregistré est présenté à l'entrée, la CMM donne : soit une adresse invalide et dans ce cas, on sait directement que le vecteur n'a pas été enregistré ; soit une adresse valide de la RAM et la détection du non appariement se fait au moment de la comparaison. Bien qu'on ne puisse pas tester le système dans tous ces états possibles (Annexe III), le processus de validation envisagé permet de certifier, a posteriori, que le système fonctionnera correctement.

Cette nouvelle architecture entraîne une légère perte de performance car elle nécessite un traitement supplémentaire des données au niveau de la RAM. Toutefois, elle permet une validation très rapide du système puisqu'il suffit de vérifier que la CMM donne la réponse souhaitée pour les vecteurs enregistrés.

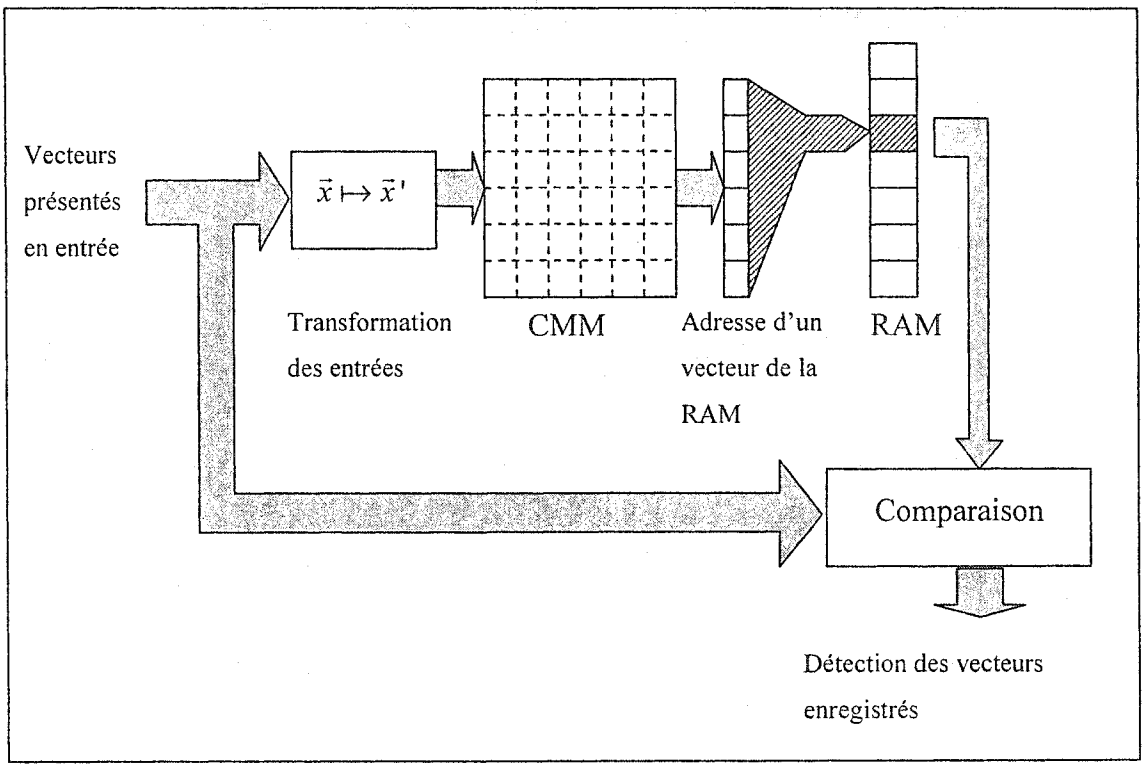


Figure 4.7 : Modèle d'une CMM pointant dans une RAM

4.4.1.2 Construction du système CMM/RAM

Le processus de construction se fait en modifiant les coefficients de la CMM d'une part et en écrivant dans la RAM d'autre part.

La première étape consiste à créer un ensemble d'adresses binaires *OUT* correspondant aux adresses d'enregistrement dans la RAM. Les adresses *OUT* sont soit des adresses réelles de la RAM, soit des adresses virtuelles qui correspondent aux adresses

d'enregistrement dans la RAM par l'intermédiaire d'une table de conversion ou LUT[7]. Dans les simulations effectuées, on utilise des adresses virtuelles de 1 à *size_in*-1 codées en binaire où *size_in* est le nombre de bits de l'entrée. La taille de vecteur de sortie *size_out* nécessaire est donc :

$$size\_out = \lceil \log_2 (size\_in - 1) \rceil \tag{4.8}$$

où la fonction $\lceil \bullet \rceil$ note la fonction de plafonnage qui arrondit à l'entier supérieur.

Lorsqu'on enregistre un vecteur $\vec{x}$, la CMM associe ce vecteur à la prochaine adresse libre et enregistre $\vec{x}$ à l'emplacement correspondant dans la RAM, ainsi que les données qui lui sont attachées. L'enregistrement de $\vec{x}$ permet d'effectuer l'appariement et les données supplémentaires permettent de déterminer les actions à réaliser en cas d'appariement. Dans le cas des règles d'un « firewall », une action peut être d'accepter ou de refuser un paquet.

4.4.1.3 Récupération des informations enregistrées

Pour déterminer si un vecteur $\vec{x}$ a été enregistré ou non, le système procède en 2 étapes :

- La CMM permet de localiser la seule adresse où peut se situer l'enregistrement correspondant au vecteur $\vec{x}$.
- La comparaison du vecteur d'entrée avec le vecteur situé à l'adresse pointée dans la RAM permet de déterminer si le vecteur d'entrée a effectivement été enregistré. En cas d'appariement on récupère alors les informations reliées au vecteur détecté.

---

[7]Acronyme de la dénomination anglaise : « Look Up Table »

4.4.2 Protocole de validation et d'évaluation du modèle

Le protocole de validation et d'évaluation du modèle consiste, dans un premier temps, à vérifier que le système fonctionne correctement. On enregistre un certain nombre d'informations supplémentaires pour pouvoir, dans un deuxième temps, évaluer la robustesse du système et les ressources matérielles nécessaires. Pour cela on cherche à connaître les ressources mémoires nécessaires pour stocker la CMM. Ces ressources sont fonction de la taille de la CMM et du nombre de bits nécessaires à coder chacun des coefficients de la CMM. Le nombre d'opérations à effectuer, quant à lui, dépend uniquement de la taille de la CMM.

Les coefficients de la CMM. Afin de pouvoir estimer correctement la taille à allouer à chaque coefficient de la CMM, les coefficients obtenus après orthogonalisation sont arrondis au dixième : comme cette précision est suffisante, il est inutile de garder des décimales supplémentaires pour les coefficients de la CMM (section 4.3.2.3).

Cas d'un test. Sur un test, on choisit le nombre d'enregistrements à stocker dans la CMM et le nombre de vecteurs de validation. Le nombre d'enregistrements est compris entre 1 et *input_size*-1. La phase de validation consiste à tester la réponse du système avec les vecteurs enregistrés : comme il a été indiqué dans la section 4.4.1, si le système donne des réponses correctes pour les vecteurs enregistrés, alors le système est valide. Les informations enregistrées sont :

- Le nombre d'erreurs commises qui doit être égal à zéro. Dans le cas contraire, le système n'est pas valide.
- La grandeur $\Delta = \|\vec{x} - \vec{y}\|_\infty$ qui est la différence maximale observée sur l'ensemble des composantes du vecteur différence pour les vecteurs enregistrés : elle permet de connaître l'erreur maximale commise en raison de l'arrondissement des coefficients de la CMM.

- Le plus grand coefficient en valeur absolue de la CMM : il permet de déterminer quelle taille doit être allouée à chacun des coefficients de la CMM.

Il est à noter que pour chaque test, une CMM est créée, celle-ci étant dépendante des vecteurs enregistrés.

Description d'une simulation. Une simulation est constituée de plusieurs séries de tests. Une série de test consiste en *input_size*-1 tests : le premier avec un enregistrement, le $2^{ème}$ avec 2 enregistrements, ..., le $i^{ème}$ avec i enregistrements... Les vecteurs enregistrés sont pris aléatoirement mais tous différents les uns des autres. Les résultats enregistrés sont déduits des résultats enregistrés pour chacun des tests. Pour chacune des 3 données enregistrées, la simulation garde le pire cas. Ceci permet d'extraire des caractéristiques indépendantes des vecteurs enregistrés dans la CMM. Ainsi, pour une simulation, les informations suivantes sont enregistrées :

- La somme du nombre d'erreurs commises sur l'ensemble de la simulation : si cette somme est non nulle, une erreur au moins a été détectée dans un des tests effectués et le modèle n'est alors pas valide.
- Le Δ[8] maximal observé sur l'ensemble de la simulation : ce Δ est la pire différence observée sur l'ensemble des simulations. Il doit être inférieur à 0,5 pour qu'aucune erreur d'arrondi ne soit commise et le plus petit possible pour assurer la meilleure robustesse.
- Le plus grand coefficient en valeur absolue sur l'ensemble de toutes les CMM observées : cette donnée permet de déterminer quelle taille doit être attribuée à chacun des coefficients de la CMM, indépendamment des vecteurs enregistrés dans la CMM.

Dans une simulation, on effectue des séries de test pour des tailles d'entrée allant de 4 à 128 bits par sauts de 4 bits. Afin d'avoir les résultats les plus significatifs, il faut faire le plus grand nombre possible de simulations.

---

[8] Ce Δ est celui décrit dans la description d'un test

4.4.3 Un modèle valide

Les résultats obtenus (Tableau 4.3) ont porté sur un ensemble de 547 simulations[9], ils permettent de vérifier que le système ne commet aucune erreur, d'évaluer la précision de la CMM et de dimensionner les coefficients de la CMM. Du nombre de bits à allouer à chaque coefficient *size_coeff*, on déduit le nombre de bits nécessaires par vecteur enregistrés *nbits_enr*. En utilisant les notations de la section 4.4.1.2 et en désignant par *nvect_max* le nombre maximal de vecteurs stockables dans la CMM, on obtient :

$$nbits\_enr = \frac{size\_in * size\_out * size\_coeff}{nvect\_max} \tag{4.9}$$

$$nbits\_enr = \frac{size\_in * \lceil \log_2 (size\_in - 1) \rceil * size\_coeff}{size\_in - 1} \tag{4.10}$$

C'est en utilisant l'équation (4.10) que l'on calcule le « Nombre de bits par enregistrement ».

---

[9] Dans un souci pratique, le paramètre d'arrêt du programme qui effectuait les simulations était un paramètre temporel et non un nombre de simulations à effectuer. Ainsi le nombre de simulations n'est pas un nombre « rond ».

Tableau 4.3 : Synthèse des résultats de simulation avec le modèle d'orthogonalisation d'une CMM pointant dans une RAM

Taille de l'entrée	Taille de la sortie	Erreur maxi--male en sortie	Coefficient maxi--mum de la CMM	Nombre de bits par coefficient	Nombre de bits par enregistrement
4	2	0,072	148,3	11	29
8	3	0,082	7436,8	17	58
12	4	0,098	12560	17	74
16	4	0,106	32979,2	19	81
20	5	0,108	51490,1	19	100
24	5	0,123	69389	20	104
28	5	0,118	1829540	25	130
32	5	0,122	75650,3	20	103
36	6	0,125	112343,9	21	130
40	6	0,133	170238,1	21	129
44	6	0,127	3323630	25	154
48	6	0,120	1831107	25	153
52	6	0,126	911164	24	147
56	6	0,138	114513	21	128
60	6	0,132	411352,5	22	134
64	6	0,135	24265440	28	171
68	7	0,128	1023463	24	171
72	7	0,135	518338,4	23	163
76	7	0,130	1681783	25	177
80	7	0,139	17084755	28	199
84	7	0,129	1427060	24	170
88	7	0,141	6106495	26	184
92	7	0,135	2964863	25	177
96	7	0,137	2626187	25	177
100	7	0,138	10078600	27	191
104	7	0,152	3309644	25	177
108	7	0,138	37389290	29	205
112	7	0,138	4245330	26	184
116	7	0,142	1833605	25	177
120	7	0,144	426880000	32	226
124	7	0,139	4842343	26	184

4.4.3.1 Un système sans erreur

Sur l'ensemble des simulations effectuées, aucune erreur n'a été commise. Ceci permet de s'assurer que l'arrondissement des coefficients de la CMM au dixième est suffisant pour ne commettre aucune erreur. Pour mieux apprécier la fiabilité du système, il faut toutefois étudier la précision de la sortie obtenue.

4.4.3.2 Précision de la réponse donnée par la CMM

La sortie désirée étant binaire, l'impératif est d'obtenir une différence inférieure à 0,5 pour chacune des valeurs de sortie. La précision obtenue est donc relativement bonne puisque la distance[10] maximale observée entre la réponse désirée et la réponse obtenue est inférieure à 0,16 (Figure 4.8). Ainsi l'arrondissement au dixième des coefficients de la CMM permet d'obtenir des résultats concluants avec une bonne marge de sécurité.

4.4.3.3 Dimensionnement des coefficients

Les simulations effectuées permettent d'évaluer la taille nécessaire à allouer à chacun des coefficients de la CMM. Avec les simulations, la taille maximale nécessaire trouvée est 32 bits pour une taille d'entrée de 120 bits (Figure 4.9) : il faut réserver en mémoire 226 bits par vecteurs enregistré soit presque deux fois la taille initiale du vecteur.

De plus, il faut être conscient que les simulations permettent de déterminer une taille minimale pour les coefficients mais n'assurent pas que cette taille sera suffisante dans tous les cas de figures. Le problème reste de savoir si les simulations effectuées englobent les pires cas ou non.

4.4.3.4 Nombre de bits par enregistrement

Pour toutes les tailles de vecteur d'entrée, le nombre de bits nécessaires pour un enregistrement est largement supérieur au nombre de bits des vecteurs d'entrée :

- Pour des tailles d'entrée inférieures à 50 bits, il faut au moins 3 fois plus de bits en mémoire que n'en compte le vecteur à stocker.
- Pour des tailles de vecteurs d'entrée de plus de 50 bits, il faut allouer en mémoire entre 2 et 3 fois la taille d'un vecteur pour le stocker dans la CMM.

---

[10] En utilisant la norme infinie

Ces résultats montrent que la méthode d'enregistrement nécessite de grosses ressources mémoires.

4.4.3.5 Ressources de calcul nécessaires

Au niveau de la CMM, les calculs à effectuer sont des additions et des multiplications. En notant respectivement *nb_add* et *nb_mult* le nombre d'additions et le nombre de multiplications à effectuer dans la CMM, *size_in* et *size_out* les dimensions de la CMM, on a :

$$nb\_add = (size\_in - 1) * size\_out \tag{4.11}$$

$$nb\_mult = size\_in * size\_out \tag{4.12}$$

On présente les résultats obtenus en fonction de la taille des vecteurs d'entrée dans le Tableau 4.4.

On constate que le nombre d'opérations à effectuer est toujours inférieur à 1000 opérations dans les simulations effectuées. Toutefois, ces opérations sont faites avec des nombres codés sur 11 à 32 bits (Tableau 4.3). Les ressources matérielles à mettre en œuvre sont donc très élevées.

Tableau 4.4 : Nombre d'opérations nécessaires avec une CMM pointant dans une RAM

Nombre de bits en entrée	Nombre de bits en sortie	Nombre d'additions	Nombre de multiplications
4	2	6	8
8	3	21	24
12	4	44	48
16	4	60	64
20	5	95	100
24	5	115	120
28	5	135	140
32	5	155	160
36	6	210	216
40	6	234	240
44	6	258	264
48	6	282	288
52	6	306	312
56	6	330	336
60	6	354	360
64	6	378	384
68	7	469	476
72	7	497	504
76	7	525	532
80	7	553	560
84	7	581	588
88	7	609	616
92	7	637	644
96	7	665	672
100	7	693	700
104	7	721	728
108	7	749	756
112	7	777	784
116	7	805	812
120	7	833	840
124	7	861	868
128	7	889	896

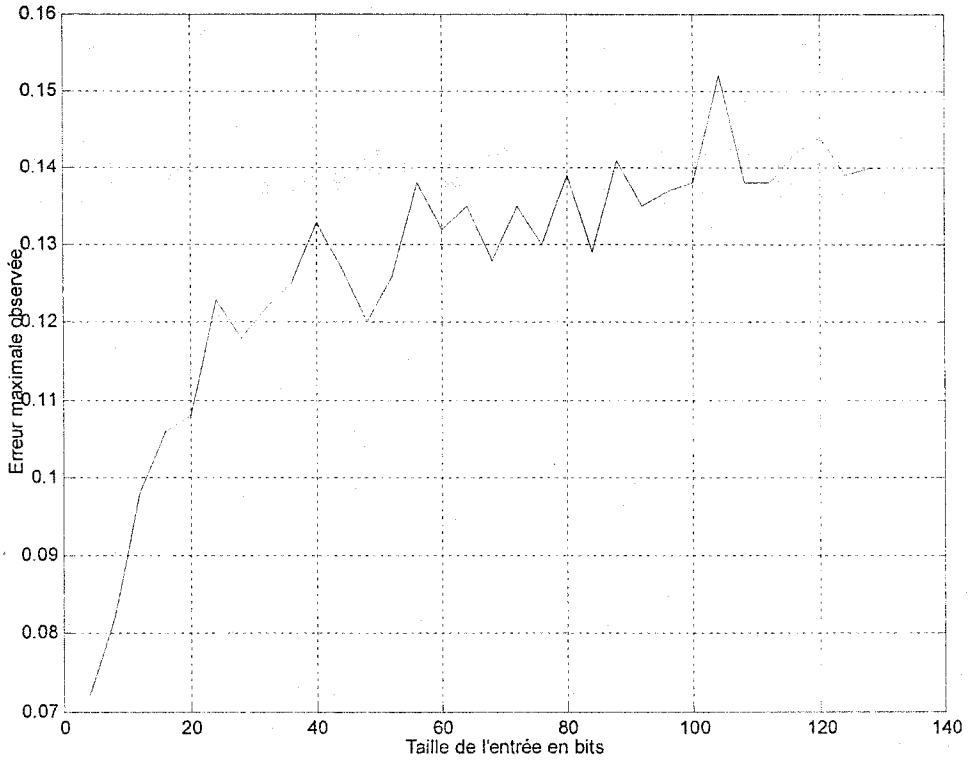


Figure 4.8 : Précision du calcul de la sortie pour des vecteurs enregistrés

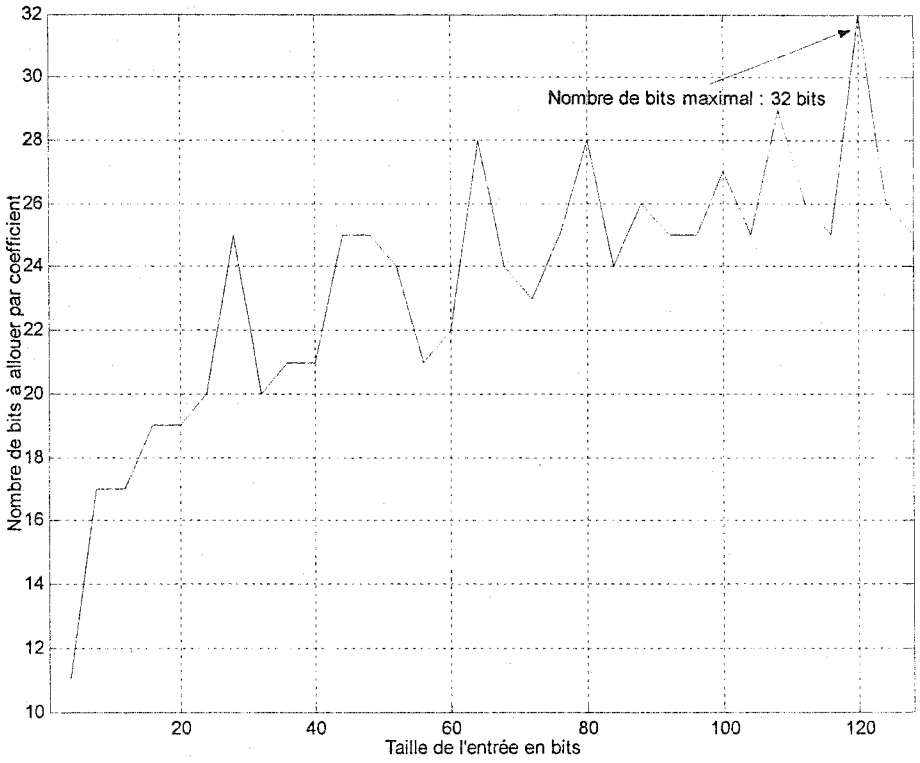


Figure 4.9 : Dimensionnement des coefficients de la CMM

4.4.4 Un système exigeant en ressources

4.4.4.1 Les incertitudes du modèle

Le problème majeur rencontré est la difficulté de dimensionner les coefficients de la CMM. En effet, les simulations effectuées fournissent une condition nécessaire (c'est-à-dire une taille minimale) quant à la taille des coefficients alors qu'il faudrait connaître une condition suffisante, c'est-à-dire une taille maximale pour les coefficients. Ces renseignements permettent néanmoins de donner une estimation des ressources mémoires nécessaires pour implanter ce modèle.

4.4.4.2 Les ressources minimales nécessaires

Si ce modèle est valide, il est très exigeant tant en matière de ressources mémoire qu'en matière de temps de calcul : il est nécessaire d'avoir des coefficients de matrice codés sur 32 bits, ce qui signifie aussi qu'il faut un espace mémoire 32 fois plus grand que l'espace mémoire nécessaire pour enregistrer les règles dans une RAM : en réalité, ce modèle exploite mal les propriétés spécifiques aux réseaux de neurones car il ne permet pas de synthétiser les données stockées. De plus, il est impossible d'assurer que les coefficients sont dimensionnés correctement. L'objectif est de trouver un système beaucoup moins exigeant en ressource qui exploite plus efficacement les propriétés des réseaux de neurones.

4.5 Randomisation des entrées sur une CMM binaire à seuil unique

La différence majeure entre les CMM binaires et les CMM étudiées précédemment se situe au niveau des coefficients de la matrice : dans les modèles précédents, les CMM étaient constituées avec des coef ficients à valeurs réelles. Dans les CMM binaires, chaque coefficient est constitué d'un seul bit, ce qui réduit l'espace mémoire nécessaire et la complexité du traitement.

La recherche par dichotomie. Dans la suite de la section, on mentionne une méthode de recherche appelée dichotomie : c'est une méthode optimisée de recherche d'un élément dans un ensemble ordonné. La dichotomie est une méthode itérative permettant de restreindre de moitié l'espace de recherche à chaque étape. Cette méthode nécessite que les N vecteurs à enregistrer soient stockés par ordre croissant dans la mémoire. L'enregistrement M situé au $(N/2)^{ème}$ emplacement est comparé à l'entrée E :

- Cas où E<M : E, s'il a été enregistré, est l'un des N/2 premiers enregistrements.
- Cas où E>M : E, s'il a été enregistré, est l'un des N/2 derniers enregistrements.

- Si E=M, alors E a été retrouvé.

On applique ce procédé sur le sous-ensemble sélectionné jusqu'à aboutir au vecteur recherché ou jusqu'à ce que le sous-ensemble inspecté ne comporte qu'un seul élément. Grâce à une complexité en $o(\log_2(N))$, la dichotomie permet de savoir rapidement si un vecteur a été enregistré ou non.

Des réseaux de neurones pour restreindre l'espace de recherche. L'idée développée consiste à utiliser les réseaux de neurones pour effectuer un prétraitement et ainsi restreindre l'espace de recherche : supposons que l'on ait enregistré 2^{10}=1024 vecteurs, il faut effectuer 10 itérations en procédant par dichotomie. Si l'on dispose d'un système permettant de sélectionner un sous-ensemble de candidats possibles, disons par exemple 2^3=8 vecteurs, il y aura 3 itérations à faire une fois le sous-ensemble trouvé, ce qui signifie qu'un prétraitement durant moins de temps que 7 itérations permettra d'améliorer les performances. Une telle idée a été développée car elle exploite la capacité que peuvent avoir les CMM à interpoler les réponses à donner.

Généralisation du modèle de prétraitement par une CMM. Grâce à ces considérations, on peut généraliser le modèle décrit dans la section 4.4 : dans ce modèle la CMM ne pointe pas vers un élément mais vers un ensemble d'éléments (Figure 4.10) dont le nombre constitue un des paramètres du système. Chaque ensemble d'éléments doit être ordonné.

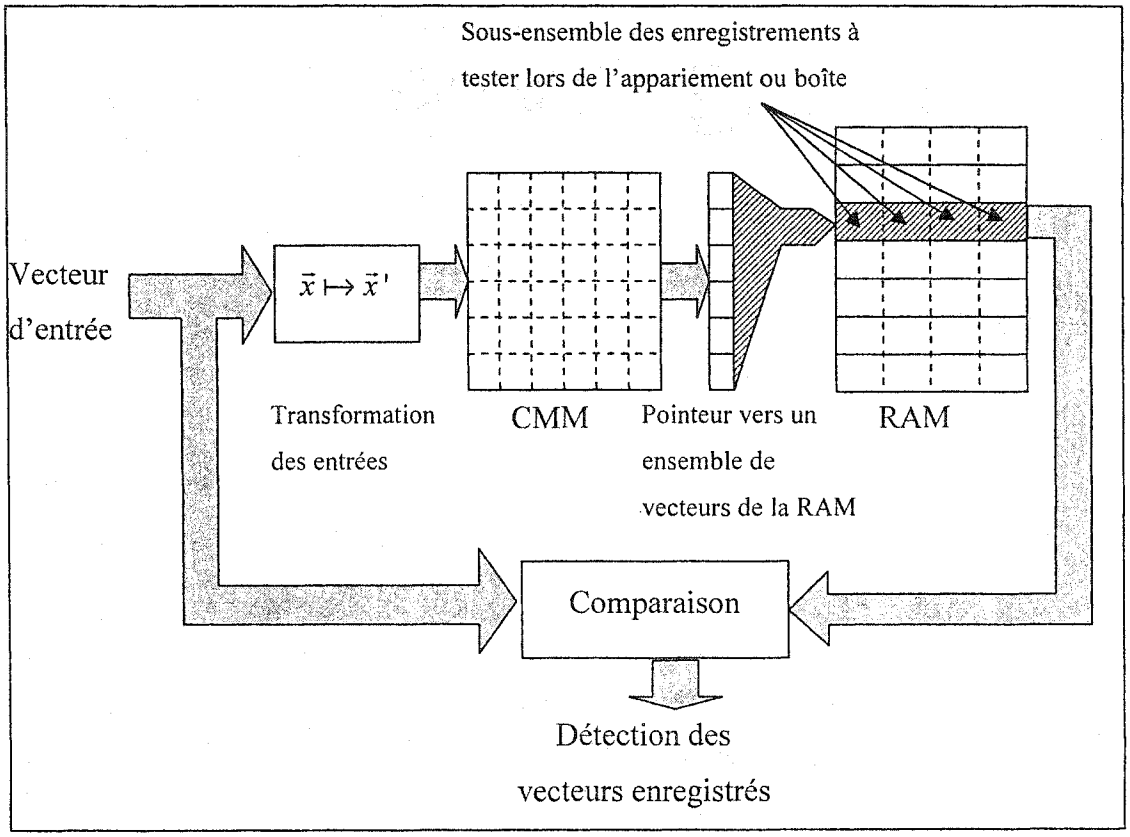


Figure 4.10 : Modèle généralisé du prétraitement par une CAM neuronale

4.5.1 Adaptation du classificateur k-NN utilisant une CMM binaire (Zhou)

4.5.1.1 Motivation du choix d'architecture

Le modèle développé par Zhou [ZHO99], permet d'utiliser une CMM à poids binaires, il est par conséquent très économique en ressources matérielles et logicielles. Dans le modèle développé par Zhou, l'objectif est de classifier des vecteurs connus ou inconnus et de donner la meilleure réponse possible en fonction des informations déjà enregistrées : le système doit donc interpréter ce qu'il a appris. Dans le cas de la détection des règles, aucune interprétation ne peut être faite : la réponse doit respecter strictement les informations enregistrées et doit pouvoir différencier ce qui est connu de

ce qui est inconnu. Il est donc nécessaire de modifier l'architecture de Zhou pour créer un système de détection de règles.

4.5.1.2 Principe de fonctionnement

Description du système. D'un point de vue général, le modèle de Zhou comporte deux étapes :

- Un prétraitement réalisé avec une CMM binaire qui permet de sélectionner un sous-espace de recherche.
- Une détermination de la classe d'appartenance en appliquant l'algorithme k-NN (section 3.3.1) au sous-ensemble sélectionné.

On peut déjà remarquer la grande similitude entre ces 2 étapes de traitement et les fonctions fondamentales de la CAM établies dans la section 2.2.

La création du système. La création du classificateur se décompose en 3 étapes :

- Regroupement des données enregistrées en sous-ensembles appelés boîtes
- Création de la CMM établissant le lien entre chaque vecteur et son sous-ensemble
- Ecriture des vecteurs à enregistrer dans les boîtes

La récupération des données. Pour retrouver les données enregistrées ou pour les interpréter, le système va sélectionner une boîte puis appliquer l'algorithme k-NN dans le sous-ensemble.

4.5.1.3 Les étapes du traitement

Le regroupement en sous-ensembles. Cette étape débute en triant les vecteurs d'entrée à enregistrer en ordre croissant. Puis les boîtes sont créées par un algorithme

d'uniformisation pour que chaque boîte[11] contienne à peu près le même nombre d'éléments. A chaque boîte est attribué un identifiant, ce qui permettra ensuite de tester le bon sous-ensemble. Il est recommandé d'avoir plus de k éléments par boîte afin que l'application de l'algorithme k-NN soit utile.

Création de la CMM binaire. En plus des impératifs propres aux CMM, une CMM binaire doit être construite avec des vecteurs « sparses », c'est-à-dire dont la plupart des composantes sont nulles. Pour parvenir à cela, Zhou utilise un algorithme de pseudo-orthogonalisation. La plupart du temps, la méthode utilisée est la méthode du N-tuple [BLE59]. Soit $\vec{b}_i$ l'ensemble des règles binaires à enregistrer et $\vec{s}_i$ les classes leur correspondant. Les classes $\vec{s}_i$ sont codées binairement afin de permettre la construction de la CMM. A partir de ces vecteurs, la CMM est construite comme telle :

$$M = \bigvee_{i=1}^{n} \vec{s}_i \vec{b}_i^T \tag{4.13}$$

Le produit utilisé est un "ET" logique et l'opérateur « $\vee$ » note un "OU" logique entre tous les produits externes $\vec{s}_i \vec{b}_i^T$. Cette construction de la CMM explique pourquoi il est nécessaire de travailler avec des vecteurs « sparses » car, dans le cas contraire, on risquerait d'obtenir une matrice M remplie de '1'.

Processus de lecture dans une CMM binaire. Le processus de lecture est similaire à celui observé dans toute CMM :

$$\vec{v} = M\vec{b} \tag{4.14}$$

Il faut noter que, dans le processus de lecture, l'addition n'est pas un "OU" logique mais bien une addition arithmétique. La multiplication utilisée est, quant à elle, toujours un "ET" logique. Une fois $\vec{v}$ calculé, on procède à son seuillage pour obtenir un vecteur binaire. Ce seuillage peut être identique ou différent pour chacune des composantes de

[11] Dans le cas de Zhou, il est fréquent que les vecteurs à enregistrer se répètent

$\vec{v}$: les deux cas de figures seront étudiés par la suite. Le vecteur binaire résultant détermine le sous-ensemble sur lequel on doit appliquer l'algorithme k-NN.

On peut mentionner que si la taille des vecteurs d'entrée est très grande, le calcul du vecteur de sortie peut se faire en plusieurs étapes : ceci revient en fait à diviser la CMM en blocs et à effectuer un produit matriciel par blocs. L'intérêt de cette possibilité est de permettre un réglage du rapport traitement en série/traitement en parallèle.

Ecriture des vecteurs à enregistrer dans les boîtes. Les vecteurs à enregistrer et la classe à laquelle ils appartiennent sont stockés dans un des emplacements de la boîte qu'on leur a attribuée.

Interprétation des informations enregistrées. Pour utiliser les informations enregistrées, on présente un vecteur connu ou inconnu à l'entrée du système. La CMM indique à quelle boîte appartient le vecteur présenté. Dans le sous-ensemble sélectionné, l'algorithme k-NN est appliqué pour déterminer à quelle classe il appartient. Cette méthode utilisant un prétraitement par une CMM permet d'obtenir une accélération d'un facteur 4 en logiciel et jusqu'à un facteur 12 en matériel par rapport à un algorithme k-NN classique implanté respectivement en logiciel et en matériel [ZHO99].

Adaptablité du modèle de Zhou. D'après les résultats obtenus avec le modèle de Zhou, on peut s'attendre également à une nette amélioration des performances en utilisant un schéma de fonctionnement du même type pour notre CAM neuronale. Mais le modèle de Zhou n'est pas intégralement retranscriptible au problème des règles d'un système coupe-feu : le traitement par k-NN n'est pas du tout adapté à ce genre de problème car il faut retrouver la règle exacte et non la règle la plus similaire. En effet, deux règles différant d'un seul bit peuvent avoir des significations très différentes. Ainsi il faut modifier l'étape correspondant à l'application de l'algorithme k-NN par un appariement consistant en une recherche par dichotomie dans le sous-ensemble sélectionné.

4.5.1.4 La transformation des entrées

Afin que le système fonctionne, il doit répondre à plusieurs impératifs :

- Les vecteurs d'entrées à apprendre doivent être orthogonaux ou pseudo-orthogonaux (section 3.4.2)
- Les vecteurs d'entrée doivent être normalisés : au sens de Hamming, cela signifie qu'ils doivent tous comporter le même nombre de '1'.
- Il faut que les vecteurs en entrée de la CMM soient « sparses » (section 4.5.1.3)

En réalité, il suffit de transformer les vecteurs d'entrée en des vecteurs « sparses » et situés suffisamment loin les uns des autres, car cela permet de retrouver correctement les vecteurs enregistrés.

Transformation A. La première transformation envisagée multiplie la taille des vecteurs par 2. Elle est tirée de [AUS98] et a été inspirée par la méthode du N-tuple [BLE59]. Dans cette transformation, le vecteur est divisé en petits vecteurs de 2 bits. Chacun des petits vecteurs va subir la transformation décrite dans le Tableau 4.5. Une fois transformés, ces petits vecteurs sont concaténés pour former le vecteur transformé.

Par exemple, le vecteur (1,0,0,0) va être divisé en (1,0) et (0,0). Chacun des petits vecteurs va être transformé, ce qui donne :

- (1,0) => (0,0,1,0)
- (0,0) => (1,0,0,0)

Le vecteur finalement obtenu est donc (0,0,1,0,1,0,0,0).

On remarque que la proportion de '1' est ¼ dans tous les cas et que la transformation réalisée normalise ainsi les vecteurs.

Tableau 4.5 : Transformation A

Vecteur d'entrée	Vecteur de sortie
00	1000
01	0100
10	0010
11	0001

Transformation B. Les règles d'un système coupe-feu disposent de parties récurrentes telles les adresses IP de destination (Annexe II). Avec de tels vecteurs, le seuil peut être impossible à ajuster si deux règles sont très proches au sens de Hamming mais que leurs sorties respectives sont très éloignées l'une de l'autre. En effet, les CMM utilisent des hypothèses de régularité : ainsi, deux entrées proches devraient donner deux sorties proches. Pour pallier à cet inconvénient, un processus de randomisation a été développé : il consiste à donner deux possibilités de codage pour chaque groupe de 2 bits (Tableau 4.6), on fera référence à cette transformation par le nom de transformation B. Pour chaque groupe de 2 bits, une des deux possibilités de codage est prise de manière aléatoire. Ainsi, un vecteur de 128 bits contient 64 groupes de 2 bits et peut ainsi être codé de 2^{64} manières différentes. Les vecteurs transformés servent de vecteurs d'entrée à la CMM. Il faut noter que cette transformation multiplie la taille des vecteurs par 4. Cependant, cette augmentation de la taille est beaucoup moins pénalisante que dans la méthode d'orthogonalisation car les coefficients de la CMM sont des nombres binaires et non des entiers ou des réels.

Tableau 4.6 : Tansformation B

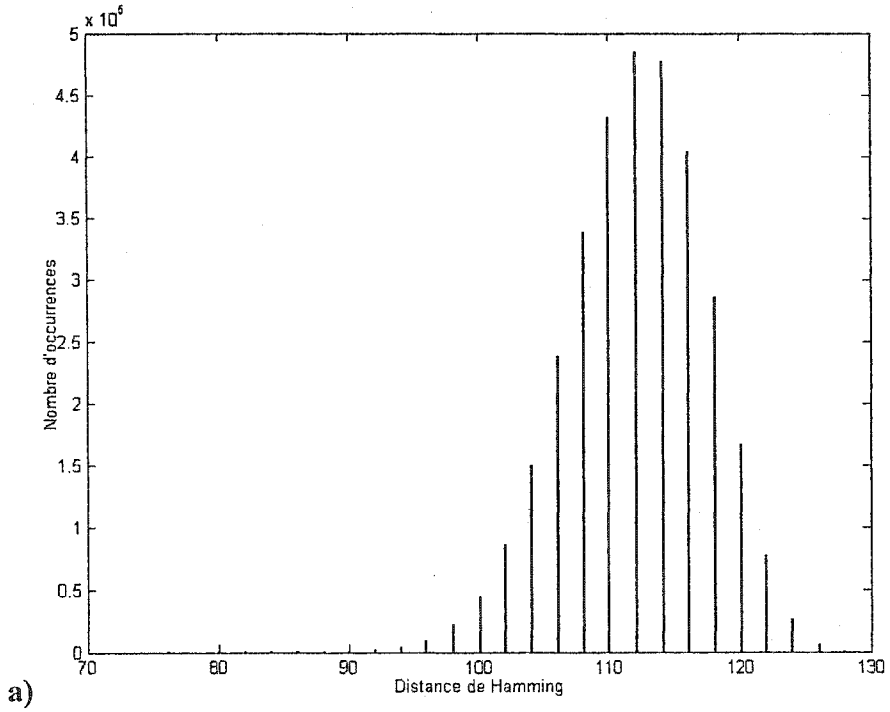
Vecteur d'entrée	Vecteur de sortie
00	10000000
	01000000
01	00100000
	00010000
10	00001000
	00000100
11	00000010
	00000001

Transformation C. Pour la lecture, la transformation des vecteurs est un peu différente de celle employée pour construire la CMM : il faut être certain de la réponse obtenue. On utilise donc les codes du Tableau 4.7 : on assure ainsi d'activer tous les bits désirés ; toutefois, ces codes activent également des bits non désirés : le réglage du seuil est par conséquent un peu plus difficile à réaliser.

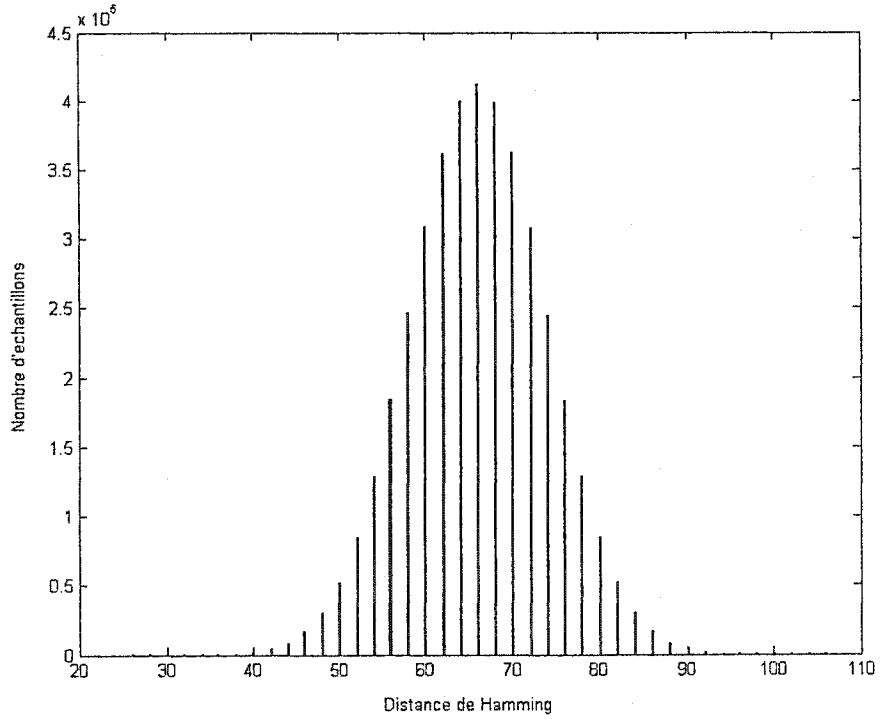
Tableau 4.7 : Tansformation C

Vecteur d'entrée	Vecteur de sortie
00	11000000
01	00110000
10	00001100
11	00000011

Vérification du fonctionnement de l'algorithme de randomisation. Les essais effectués avec des vecteurs aléatoires (Figure 4.11a) ou des vecteurs proches au sens de Hamming (Figure 4.11b) montrent la distribution des distances de Hamming entre les vecteurs transformés. Les vecteurs dits « proches au sens de Hamming » sont générés grâce au procédé suivant : on prend un vecteur aléatoire, puis on crée tous les vecteurs qui ont un bit de différence avec lui. Par conséquent, si on considère l'ensemble des vecteurs obtenus, ils sont tous distants de 1 ou de 2 les uns des autres. La distribution gaussienne des distances de Hamming étant caractéristique d'une répartition uniforme des vecteurs sur l'espace, on constate que la transformation B répartit uniformément les vecteurs. Dans les deux cas, les distributions des distances après transformation sont des gaussiennes pour lesquelles les distances moyennes entre vecteurs sont respectivement de 112 et de 64 bits : on peut ainsi considérer que les vecteurs obtenus après transformation sont suffisamment éloignés les uns des autres pour pouvoir être traités correctement dans la CMM.



a)



b)

Figure 4.11 : Distribution de la distance de Hamming entre 129 vecteurs de 512 bits a) issus de vecteurs aléatoires de 128 bits b) issus de la transformation de vecteurs de 128 bits proches au sens de Hamming

4.5.1.5 La création de la CMM

Dans cette section on note $\{\vec{x}_i\}_{i\in[1,d]}$ les d vecteurs d'entrée à enregistrer et $\{\vec{y}_i\}_{i\in[1,d]}$ les sorties leur correspondant au niveau de la CMM. Tous ces vecteurs sont des vecteurs binaires.

La transformation des vecteurs d'entrée. Les vecteurs d'entrée $\{\vec{x}_i\}_{i\in[1,d]}$ sont transformés en vecteurs $\{\vec{x}'_i\}_{i\in[1,d]}$ par la transformation B. Ceci permet de créer la CMM à partir de vecteurs « sparses » : la taille des vecteurs est donc multipliée par 4. Pour des vecteurs d'entrée de 128 bits de long, les vecteurs en entrée de la CMM compteront 512 bits.

Le choix des vecteurs de sortie. Comme il a été mentionné précédemment (section 4.5.1.3), les vecteurs de sortie sont des vecteurs binaires. Chaque sous-ensemble doit correspondre à une sortie unique : on peut, a priori, envisager toutes les sorties possibles. Cependant, en fixant un seuil unique, la sortie est contrainte à être un vecteur ne contenant qu'un seul '1' : en effet, si le vecteur de sortie contient plusieurs '1', il est parfois impossible de régler le seuil. Les vecteurs de sortie sont donc les vecteurs binaires $\{\vec{y}_i\}_{i\in[1,d]}$ pour lesquels chaque vecteur $\vec{y}_i$ comporte un '1' sur sa $i^{\text{ème}}$ composante seulement.

4.5.1.6 La création des boîtes

Après avoir généré la CMM, on crée les boîtes et on enregistre dans chacune d'entre elles les éléments qu'on lui a attribués. Dans chacune de ces boîtes, les éléments doivent être ordonnés afin de pouvoir procéder à une recherche par dichotomie lors de la phase d'appariement.

4.5.2 Recherche de la capacité maximale de la CMM

Le protocole de simulation est très similaire à celui adopté dans la section 4.4.2. La seule différence se situe au niveau des informations enregistrées lors de la simulation. Afin de trouver la capacité maximale de la CMM, les tailles de boîtes pour lesquelles le seuillage est possible sont enregistrées. En effet, il n'est pas toujours possible de régler correctement le seuil étant donné le mode d'écriture utilisé (section 4.4.1.2). Si trop de vecteurs sont enregistrés, il y a un risque de superposition et, à partir d'un certain point, il sera impossible d'extraire l'information spécifique à une entrée. Le cas limite est en fait une CMM où tous les coefficients valent '1'.

4.5.2.1 Procédé d'ajustement automatique du seuil

Méthode d'ajustement. Dans cette simulation, les vecteurs transformés par le processus de randomisation sont enregistrés dans la CMM binaire (section 4.5.1.3). Ensuite, on règle le seuil en adoptant les étapes suivantes :

- On transforme les vecteurs d'entrée $\{\vec{x}_i\}_{i\in[1,d]}$ par la transformation C en vecteurs $\{\vec{x}'_i\}_{i\in[1,d]}$.
- Ces vecteurs sont passés dans la CMM pour obtenir les vecteurs à valeurs entières $\left\{\hat{\vec{Y}}_i\right\}_{i\in[1,d]}$.
- Les vecteurs $\left\{\hat{\vec{Y}}_i\right\}_{i\in[1,d]}$ sont ensuite seuillés pour obtenir les vecteurs binaires $\left\{\hat{\vec{y}}_i\right\}_{i\in[1,d]}$.
- Ces vecteurs binaires sont comparés aux vecteurs souhaités $\{\vec{y}_i\}_{i\in[1,d]}$.
- Le réglage du seuil se fait ensuite par essai erreur (Figure 4.12) jusqu'à ce que l'on ait $\hat{\vec{y}}_i = \vec{y}_i$ pour tout *i*.

- Si jamais tous les seuils ont été testés, la recherche s'interrompt et le système signale qu'aucun seuil ne convient.

Modification du seuil. La modification du seuil pourrait se faire par simple incrémentation en initialisant le seuil à 1. Mais cette méthode est relativement lente, sa complexité est en *o(n)*, si on note *n* la taille des vecteurs d'entrée. Afin d'accélérer le réglage du seuil, on utilise un algorithme de complexité $o(log_2(n))$.

Sachant que les vecteurs d'entrée sont soumis à la transformation B lors de la création de la CMM, le nombre maximal de '1' obtenus en sortie est *n*/4. Le seuil est ainsi initialisé à la puissance de 2 directement supérieure à $\frac{n/4}{2} = n/8$ soit $S_0 = 2^{\lceil \log_2(n/8) \rceil}$. Ensuite, la modification du seuil sera fonction des vecteurs différence $\left\{ \hat{\vec{y}}_i - \vec{y}_i \right\}_{i \in [1,d]}$. Si la modification du seuil à l'itération suivante est supérieure ou égale à 1[12] :

- *Cas 1.* Si toutes les composantes de tous les vecteurs différence sont nulles, alors $\hat{\vec{y}}_i = \vec{y}_i$ pour tout $i \in [1,d]$. Le seuil utilisé convient et l'ajustement s'interrompt.
- *Cas 2.* Si l'une des composantes de n'importe lequel de ces vecteurs différence est 1, un des $\hat{\vec{y}}_i$ au moins comporte un '1' au lieu d'un '0' : le seuil est trop bas et il faut le rehausser au pas suivant.
- *Cas 3.* Si aucune des composantes de tous ces vecteurs différence est 1 mais que l'une des composantes de n'importe lequel de ces vecteurs différence est -1, un des $\hat{\vec{y}}_i$ au moins comporte un '0' au lieu d'un '1' : le seuil est trop haut et il faut le baisser au pas suivant.

---

[12] Si elle est inférieure à un, il faut arrêter la recherche car tous les seuils possibles ont été testés

On peut donc définir la modification du seuil par récurrence :

$S_0 = 2^{\lceil \log_2(n/8) \rceil}$

$\Delta S_0 = \frac{S_0}{2}$

Pour tout $k \geq 0$

- Si on se trouve dans le cas 1

  Le seuil est correct

- Si on se trouve dans le cas 2

  $S_{k+1} = S_k + \Delta S_k$ et $\Delta S_{k+1} = \frac{\Delta S_k}{2}$

- Si on se trouve dans le cas 3

  $S_{k+1} = S_k - \Delta S_k$ et $\Delta S_{k+1} = \frac{\Delta S_k}{2}$

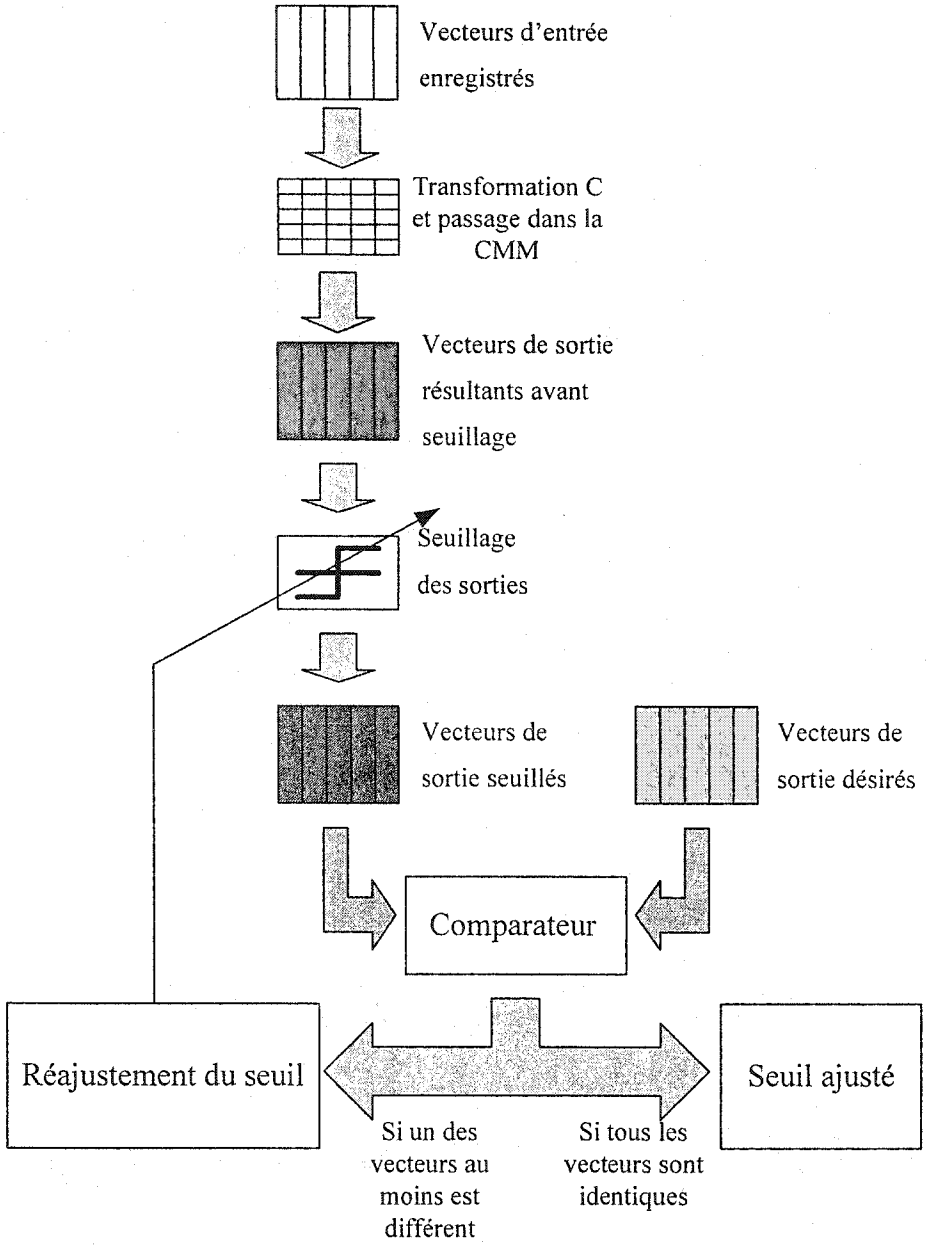


Figure 4.12 : Processus d'ajustement du seuil

Plusieurs essais pour trouver un seuil. L'ajustement du seuil est parfois impossible car les vecteurs utilisés pour créer la CMM sont trop proches les un des autres. Toutefois, comme la transformation B est partiellement aléatoire, renouveler le processus décrit sur la Figure 4.12 permet d'obtenir des vecteurs d'entrée de la CMM différents d'un essai à

l'autre. Faire un grand nombre de tentatives augmente les chances d'obtenir un ensemble vecteurs $\{\vec{x}'_i\}_{i\in[1,d]}$ assez distants les uns des autres pour permettre l'ajustement du seuil. Dans les simulations présentées, le système peut effectuer jusqu'à 200 essais pour trouver une CMM permettant de régler le seuil correctement.

4.5.2.2 Informations enregistrées en vue du dimensionnement de la CMM

Objectif des simulations. Les simulations ont pour but de déterminer, indépendamment du nombre d'enregistrements, la taille de boîte optimale. Pour cela, les simulations vont fournir, en fonction de la taille des boîtes, le nombre de vecteurs qui peuvent être enregistrés. Ceci permettra de déterminer la capacité maximale du système en fonction de la taille de boîtes utilisée.

Choix des échantillons de test. Dans toutes les simulations, la taille des vecteurs d'entrée est de 128 bits. Les vecteurs d'entrée enregistrés sont pris aléatoirement.

Description d'un test. Un test consiste à choisir un nombre d'enregistrements et à tester pour quelles tailles de boîtes le système réussit à trouver un seuil de fonctionnement. Pour limiter le temps de calcul, les tailles de boîtes testées sont les puissances de 2 inférieures au nombre d'enregistrements. Un tel choix de tailles des boîtes est adopté compte tenu de la méthode d'appariement utilisée dans laquelle la recherche est effectuée par dichotomie.

Description d'une simulation. Les simulations permettent d'effectuer des tests pour un nombre d'enregistrements allant de 8 à 256 par pas de 8. De nombreuses simulations sont réalisées pour chaque nombre d'enregistrements afin d'obtenir des résultats significatifs.

Informations enregistrées. Dans cette simulation, on veut connaître le nombre maximal d'éléments que peut enregistrer le système pour une taille de boîte donnée. Ainsi, pour chaque taille de boîte testée, on enregistre le nombre maximal d'enregistrements possibles, c'est-à-dire le nombre maximal d'enregistrements pour lequel on réussit à régler les seuils dans toutes les simulations.

4.5.3 Un système validé

4.5.3.1 Synthèse des résultats

Les résultats présentés (Tableau 4.8) ont été obtenus avec 40 simulations. Le nombre de bits par enregistrement est le nombre de bits de la CMM divisé par le nombre de vecteurs enregistrés. Les vecteurs d'entrée de la CMM ont 4*128=512 bits de long en raison de la transformation B et la largeur de la CMM est égale au nombre de boîtes.

Tableau 4.8 : Synthèse des résultats obtenus avec le modèle de randomisation avec CMM à seuil unique

Taille des boîtes	Nombre d'enregis-trements possibles	Nombre de boîtes	Nombre de bits par enregistrement
1	256	256	512
2	256	128	256
4	256	64	128
8	48	6	64
16	24	2	43

On constate que l'on peut enregistrer un grand nombre d'échantillons avec des tailles de boîtes inférieures ou égales à 4. Pour les boîtes de tailles 8 ou 16, on divise respectivement par 6 et par 2 la taille de l'espace à explorer. Dans les deux derniers cas, l'efficacité du prétraitement n'est pas assez bonne pour obtenir une amélioration significative des performances par rapport à une simple dichotomie. D'autre part, on

pourrait s'attendre à avoir un nombre de bits par enregistrement égal à 32 avec des boîtes de 16 éléments, ce qui n'est pas le cas. En fait, la « régularité » dans la diminution du nombre de bits par enregistrement lorsque la taille des boîtes augmente est seulement accidentelle. Et dans le cas des boîtes de 16 éléments, le nombre de bits par enregistrement est supérieur aux 32 bits attendus car il n'est pas possible de remplir entièrement toutes les boîtes.

Pour les tailles de boîte inférieures à 4, on constate que le coût en nombre de bits par enregistrement est supérieur ou égal au coût occasionné par un simple enregistrement dans une RAM, il présente donc un intérêt limité en terme de coût.

4.5.3.2 Nombre d'opérations à effectuer

Les résultats présentés ci-dessus permettent de calculer le nombre d'opérations nécessaires à calculer le vecteur de sortie de la CMM (Tableau 4.9). Le calcul du nombre d'additions et d'opérations "ET" sont respectivement identiques au calcul du nombre d'additions et de multiplications décrits dans la section 4.4.3.5. Les additions utilisées sont en fait des incrémentations et ne nécessitent pas de ressources matérielles élevées. La taille des vecteurs en entrée de la CMM est égale à 512 bits.

Tableau 4.9 : Nombre d'opérations dans une CMM binaire à seuil unique

Taille des boîtes	Taille de la sortie	Nombre d'additions	Nombre d'opérations "ET"
1	256	130816	131072
2	128	65408	65536
4	64	32704	32768
8	6	3066	3072
16	2	1022	1024

On constate qu'il y a avantage à avoir des boîtes de 4 éléments afin de diminuer le nombre d'opérations à effectuer tout en restreignant efficacement l'espace de recherche. Toutefois les opérations à effectuer ici, contrairement aux opérations effectuées dans les modèles d'orthogonalisation, sont des opérations très simples –opérations logiques et incrémentation d'un compteur- qui ne nécessitent que peu de ressources de calcul.

4.5.4 Un modèle perfectible

Les limites rencontrées par ce modèle, en terme de capacité mémoire et de ressources nécessaires, est causé par 2 paramètres : la méthode de seuillage du vecteur de sortie de la CMM et le choix de la sortie à enregistrer.

4.5.4.1 Le choix du seuil

Le problème de la méthode à seuil unique est qu'elle exige un réglage très précis, qui n'est pas toujours réalisable. En effet, si les '1' sont, normalement, distribués uniformément dans la CMM, il n'est pas évident que la somme maximale soit identique pour chacun des bits de la sortie. Toutefois, les résultats montrent qu'il est possible de stocker les informations relatives aux vecteurs enregistrés avec un coût inférieur au coût de stockage dans une RAM pour des tailles de boîtes d'au moins 8 enregistrements. Il faut néanmoins améliorer le modèle : le travail réalisé avec une taille de boîte égale à 8 est une réduction d'un facteur 6 de l'espace de recherche. Si on tient compte du coût occasionné par le prétraitement de la CMM en terme de temps de calcul, on constate qu'on ne peut pas améliorer significativement la vitesse de recherche comparativement à une recherche par dichotomie.

4.5.4.2 Le choix de la sortie à enregistrer

Comme il a été vu dans la section 4.5.1.5, la taille de la sortie est égale au nombre d'enregistrements : le seuil ne peut être réglé s'il y a plus d'un '1' dans le vecteur de sortie. Or cette contrainte oblige l'utilisation de vecteurs de sortie de grande taille. Ainsi, un réglage du seuil indépendant pour chacun des bits de la sortie doit permettre d'assouplir cette contrainte. Ainsi, il est possible d'utiliser des vecteurs de sortie plus petits et de disposer ainsi d'une architecture plus économique.

4.6 RANDOMISATION DES ENTRÉES SUR UNE CMM BINAIRE MULTISEUIL

Le modèle de randomisation des entrées sur une CMM binaire multiseuil est quasiment identique au modèle à seuil unique. La seule différence se situe au niveau de l'ajustement du seuil. En effet, dans le modèle multiseuil, le seuillage se fait indépendamment pour chacun des bits du vecteur de sortie de la CMM : on a donc un vecteur de seuils à calculer. L'objectif des simulations réalisées en utilisant une stratégie multiseuil est de quantifier les avantages et les inconvénients qu'apporte cette approche comparativement à une approche à seuil unique.

4.6.1 Modifications à apporter au modèle à seuil unique

4.6.1.1 Motivation d'une approche multiseuil

Avec un seuil pour chacun des bits de sortie, on peut accéder à un réglage beaucoup plus fin du système. En effet, la contrainte pour les vecteurs de sortie est beaucoup moins forte que pour les vecteurs d'entrée puisque le caractère « sparse » recherché pour les vecteurs d'entrée a pour origine le besoin d'avoir des vecteurs d'entrée quasiment orthogonaux. On peut remarquer que cette solution est obligatoirement plus performante que la méthode utilisant un seuil unique qui est un cas particulier de réglage multiseuil.

Les simulations doivent permettre de déterminer s'il est avantageux de mettre en œuvre un tel seuillage.

4.6.1.2 Des sorties codées sur moins de bits

On peut choisir les vecteurs de sortie comme on veut à condition que l'on puisse régler les seuils par la suite. En raison de la méthode de création de la CMM binaire qui superpose les enregistrements, l'ajustement des seuils est d'autant plus facile que le nombre de '1' dans les vecteurs de sortie est petit. Si les vecteurs de sortie comptent beaucoup de '1', la CMM va avoir rapidement la majorité de ses coefficients à '1', ce qui signifie qu'il y aura un grand recouvrement des informations stockées et, qu'à la limite, il sera impossible de retrouver les informations enregistrées.

Cependant pour une taille de vecteurs de sortie donnée, le nombre de boîtes adressables grandit si on augmente le nombre de '1' autorisés dans le vecteur binaire de sortie. Il faut donc rechercher le nombre optimal de '1' autorisés dans le vecteur de sortie qui permette de régler les seuils de la CMM dans tous les cas de figure. Dans l'approche multiseuil étudiée, on peut utiliser des vecteurs de sortie possédant une ou deux composantes fixées à '1'. On réduit ainsi la taille des vecteurs de sortie et, par conséquent, la taille de la CMM.

Soit *nb_boites* le nombre de boîtes et k le nombre de bits de la sortie, on cherche à exprimer k en fonction de *nb_boites*.

Si l'on dispose de k bits de sortie, alors le nombre de boîtes maximal nb_boites_{max} adressables avec un ou deux '1' dans le vecteur de sortie est :

$$\begin{aligned} nb\_boites_{\max} &= \binom{1}{k} + \binom{2}{k} = k + \frac{k(k-1)}{2} \\ &= \frac{1}{2}k^2 + \frac{1}{2}k \end{aligned} \tag{4.15}$$

Le k idéal pour adresser nb_boites_{max} boîtes est celui qui résout l'équation du second degré (4.15).

k étant un nombre entier, on obtient finalement :

$$k = \left\lceil \frac{\sqrt{8nb\_boites_{\max} + 1} - 1}{2} \right\rceil \tag{4.16}$$

Ainsi, le nombre de boîtes adressables en fonction du nombre de bits de sortie se comporte comme une fonction « racine carrée » (Figure 4.13), sa complexité est $o\left(\sqrt{n}\right)$.

Le Tableau 4.10 montre qu'il est économique d'adresser un grand nombre de boîtes avec ce modèle.

Tableau 4.10 : Nombre de boîtes adressables suivant la taille du vecteur de sortie de la CMM

Nombre de bits de sortie	Nombre de boîtes
1	1
2	2 à 3
3	4 à 6
4	7 à10
5	11 à15
6	16 à 21
7	22 à 28
8	29 à 36
9	37 à45
10	46 à 55
11	56 à66
12	67 à 78
13	79 à 91
14	92 à 105
15	106 à 120
16	121 à 136
17	137 à 153
18	154 à 171
19	172 à190
20	191 à 210
21	211 à 231
22	232 à 253
23	254 à plus de 256

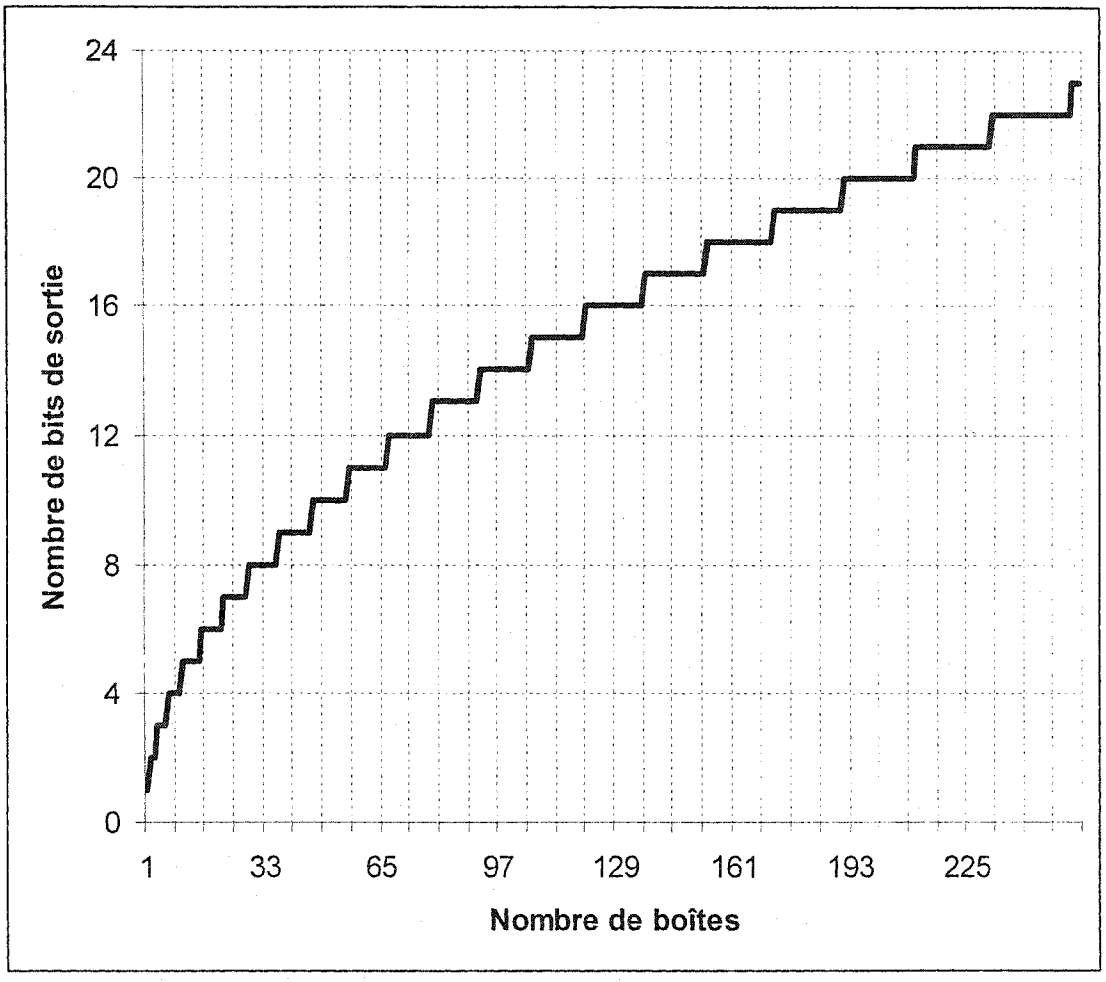


Figure 4.13 : Evolution du nombre de boîtes adressables suivant la taille du vecteur de sortie de la CMM

4.6.2 Méthode de recherche de la capacité maximale du système

L'objectif des simulations est de déterminer, pour chacune des tailles de boîtes testées, le nombre maximal de vecteurs que le système peut enregistrer. Disposant de ces données, on peut calculer le nombre de bits nécessaires pour stocker un vecteur et le nombre d'opérations à effectuer.

4.6.2.1 Principe d'ajustement des seuils

Dans ce modèle, le processus d'ajustement du seuil décrit dans la section 4.5.2.1 est appliqué indépendamment pour chacun des bits du vecteur de sortie de la CMM. On considère que le système a réussi à régler les seuils si tous les vecteurs d'entrée enregistrés donnent un vecteur de sortie exact. Le processus d'ajustement est un peu plus complexe que celui adopté dans le modèle à seuil unique et nécessite, par conséquent, un temps de calcul un peu plus long.

4.6.2.2 Méthode de validation

La méthode de validation employée est en tout point identique à celle décrite dans la section 4.5.2. Grâce à elle, on pourra déterminer quelles tailles de boîtes sont utilisables et comparer ces résultats à ceux obtenus avec le modèle utilisant un seuil unique.

4.6.3 Une amélioration du modèle à seuil unique

Les résultats obtenus donnent sur le nombre d'enregistrements possibles en fonction de la taille des boîtes. Pour chaque taille de boîte, 71[13] simulations ont été réalisées. Les résultats obtenus sont présentés dans le Tableau 4.11.

---

[13] Dans un souci pratique, le paramètre d'arrêt du programme qui effectuait les simulations était un paramètre temporel et non un nombre de simulations à effectuer. Ainsi le nombre de simulations n'est pas un nombre « rond ».

Tableau 4.11 : Synthèse des résultats obtenus dans les simulations du modèle de randomisation des entrées avec une CMM à réglage multiseuil

Taille des boîtes	Nombre d'enregis-trements possibles	Nombre de boîtes	Nombre de bits de sortie	Nombre de bits par enregistrement
1	112	112	15	69
2	64	32	8	64
4	32	8	4	64
8	32	4	3	48
16	32	2	2	32

On constate que l'on peut enregistrer d'autant plus de vecteurs que la taille des boîtes est petite. Ce résultat est intéressant dans la mesure où les configurations utilisant de petites tailles de boîtes permettent le prétraitement le plus efficace. Ici, une taille de boîte de 1 permet une réduction de l'espace de recherche d'un facteur 112 alors, qu'à l'opposé, une taille de boîte de 16 ne permet de réduire l'espace de recherche que de moitié.

D'autre part, on constate que les vecteurs de sortie utilisés permettent de réduire énormément le nombre de bits nécessaires au stockage d'un vecteur par rapport à une approche utilisant un seuil unique.

4.6.4 La recherche d'un compromis coût-performance

Les résultats obtenus montrent que le compromis coût-performance à réaliser est fortement dépendant de la taille des boîtes : en effet, la taille des boîtes influence à la fois la taille de la CMM et le nombre d'opérations à effectuer dans la phase d'appariement.

4.6.4.1 L'influence de la taille des boîtes

Les résultats des simulations montrent que le nombre d'enregistrements pour lesquels la CMM peut être correctement configurée est très dépendant de la taille et du nombre des boîtes :

- En effet dans le cas extrême d'une boîte regroupant tous les enregistrements, un seul bit de sortie est nécessaire et un seuil à zéro permettra la configuration de la CMM. Dans ce cas non étudié, le prétraitement est inutile puisqu'il ne restreint aucunement l'espace de recherche.
- Dans le cas extrême où chaque boîte ne contient qu'un élément, on peut adresser 112 boîtes. Cette méthode est la plus performante puisqu'une seule comparaison est nécessaire dans la phase d'appariement.
- Dans les cas intermédiaires, il existe un compromis à faire entre la taille de boîtes et le nombre de boîtes, ce qui influencera, par la suite, le coût et les performances du système.

4.6.4.2 Le coût en ressources

Le coût en ressource de la CMM se fait à partir du calcul du nombre de bits nécessaires en sortie (Tableau 4.10) et de la capacité maximale d'enregistrement (Tableau 4.11). Ceci permet de déterminer le nombre de bits nécessaires par enregistrement et le nombre d'opérations binaires à effectuer en fonction de la taille des boîtes. Les additions mentionnées ci-dessous sont toujours des incréments et sont, par conséquent, peu coûteuses en ressources matérielles.

Taille de boîte de 1. Quand la taille des boîtes est 1, le nombre maximal d'enregistrements possibles est 112, ce qui nécessite 15 bits en sortie : la CMM a une taille de 512x15 bits. Le coût en mémoire est donc de 69 bits par enregistrement et le nombre d'opérations est :

- 7680 opérations "ET" soit 68,6 par enregistrement
- 7665 additions soit 68,4 par enregistrement
- 15 comparaisons entières
- 1 comparaison dans la phase d'appariement

Il faudrait 9 architectures de ce type pour traiter 1000 enregistrements.

Taille de boîte de 2. Quand la taille des boîtes est 2, le nombre maximal d'enregistrements possibles est 64, ce qui nécessite 8 bits en sortie : la CMM a une taille de 512x8 bits. Le coût en mémoire est donc de 64 bits par enregistrement et le nombre d'opérations est :

- 4096 opérations "ET" soit 64,0 par enregistrement
- 4088 additions soit 63,9 par enregistrement
- 8 comparaisons entières
- 2 comparaisons dans la phase d'appariement

Il faudrait 13 architectures de ce type pour traiter 1000 enregistrements.

Taille de boîte de 4. Quand la taille des boîtes est 4, le nombre maximal d'enregistrements possibles est 32, ce qui nécessite 4 bits en sortie : la CMM a une taille de 512x4 bits. Le coût en mémoire est donc de 64 bits par enregistrement et le nombre d'opérations est :

- 2048 opérations "ET" soit 64,0 par enregistrement
- 2044 additions soit 63,9 par enregistrement
- 4 comparaisons entières
- 3 comparaisons dans la phase d'appariement

Il faudrait 32 architectures de ce type pour traiter 1000 enregistrements.

Taille de boîte de 8. Quand la taille des boîtes est 8, le nombre maximal d'enregistrements possibles est 32, ce qui nécessite 3 bits en sortie : la CMM a une taille

de 512x3 bits. Le coût en mémoire est donc de 48 bits par enregistrement et le nombre d'opérations est :

- 1536 opérations "ET" soit 48 par enregistrement
- 1533 additions soit 47,9 par enregistrement
- 3 comparaisons entières
- 4 comparaisons dans la phase d'appariement

Il faudrait 32 architectures de ce type pour traiter 1000 enregistrements.

Taille de boîte de 16. Quand la taille des boîtes est 16, le nombre maximal d'enregistrements possibles est 32, ce qui nécessite 2 bits en sortie : la CMM a une taille de 512x2 bits. Le coût en mémoire est donc de 32 bits par enregistrement et le nombre d'opérations est :

- 1024 opérations "ET" soit 32 par enregistrement
- 1022 additions soit 31,9 par enregistrement
- 2 comparaisons entières
- 5 comparaisons dans la phase d'appariement

Il faudrait 32 architectures de ce type pour traiter 1000 enregistrements.

Le Tableau 4.12 donne le nombre d'additions et d'opérations "ET" nécessaires en fonction de la taille des boîtes.

Tableau 4.12 : Nombre d'opérations dans une CMM binaire multiseuil

Taille des boîtes	Taille de la sortie	Nombre d'additions	Nombre d'opérations "ET"
1	15	7665	7680
2	8	4088	4096
4	4	2044	2048
8	3	1533	1536
16	2	1022	1024

Si le coût semble diminuer lorsqu'on augmente la taille des boîtes, le travail réalisé, comparé au même travail réalisé avec une simple dichotomie est de moins en moins avantageux. Dans le cas d'une taille de boîtes de 1, on réduit d'un facteur 112 l'espace de recherche. Ainsi le travail effectué équivaut à 7 itérations d'une dichotomie (section 4.5). Dans le cas d'une taille de boîtes de 16, on réalise une catégorisation en 2 classes, ce qui équivaut à une seule itération dans une dichotomie classique. Il est donc préférable d'utiliser de petites tailles de boîtes.

4.7 CONCLUSION

Parmi les modèles étudiés, le plus avantageux est celui utilisant une CMM binaire multiseuil. On explique par la suite comment régler les paramètres de ce système.

4.7.1 Les avantages des petites tailles de boîtes

Il est avantageux, contrairement à ce qu'il pourrait sembler au premier abord, de travailler avec de petites tailles de boîtes : en effet; les petites tailles de boîte permettent une division de l'espace en un grand nombre de sous-espaces même si le coût occasionné est un peu plus élevé que pour de grandes tailles de boîtes. Les petites tailles de boîtes sont avantageuses car elle permettent de réaliser un travail de présélection beaucoup plus important. Pour déterminer quelle taille de boîtes est la plus avantageuse, il serait possible de créer un indice qui prendrait en compte le quotient du nombre d'opérations sur le nombre de dichotomies nécessaires pour effectuer le même travail mais un tel indice serait aussi fortement dépendant d'autres critères tel le choix d'implantation.

4.7.2 La taille des boîtes dépend du compromis coût-performance recherché

La détermination de la taille des boîtes va se faire, non seulement en fonction des résultats obtenus précédemment (section 4.7.1), mais aussi du degré de parallélisation envisagé. En effet, les CMM binaires permettent autant une implantation fortement parallèle si l'on dispose des ressources matérielles adéquates, qu'une implantation favorisant un traitement en série, si les ressource matérielles sont faibles. Ainsi, les contraintes matérielles et logicielles doivent permettre de trouver une configuration optimale en terme de coût et de performance suivant le choix d'architecture.

CHAPITRE 5

SYNTHÈSE DES RÉSULTATS

5.1 INTRODUCTION

L'étude des différentes architectures envisagées a pour objectif de dégager les points forts d'une CAM neuronale. Ces points forts permettront de déterminer pour quelles applications de telles architectures sont avantageuses. Les résultats obtenus dans les simulations permettent d'évaluer quel est le meilleur modèle parmi les cinq modèles testés. Les informations relatives à la taille des CMM et aux ressources de calculs dont elles ont besoin permettent de donner les premiers éléments de comparaison entre une CAM neuronale et une CAM « traditionnelle ».

5.2 ÉTUDE COMPARÉE DES MODÈLES ÉTUDIÉS

Dans cette section, on présente en parallèle les résultats obtenus avec les différents modèles afin de pouvoir comparer leurs caractéristiques. On rappelle d'abord quels modèles ont été validés puis on présente les résultats de dimensionnement des CMM pour les modèles valides. Ce dimensionnement permet de déterminer les ressources mémoire et les ressources matérielles nécessaires. Enfin, on explique les raisons pour lesquelles le modèle de CMM binaire multiseuil semble être le meilleur.

5.2.1 Synthèse des résultats de validation

La première phase de chaque simulation consistait à valider les architectures envisagées. Les résultats des tests de validation donnés dans les sections 4.2.3, 4.3.3, 4.4.3, 4.5.3 et 4.6.3 sont résumés dans le Tableau 5.1.

Tableau 5.1 : Synthèse des résultats de validation

	Valide	Apparemment valide	Non valide
Orthogonalisation des vecteurs complémentés			X
Orthogonalisation par recodage caractéristique de la composante		X	
Orthogonalisation d'une CMM pointant dans une RAM	X		
Randomisation des entrées sur une CMM binaire à seuil unique	X		
Randomisation des entrées sur une CMM binaire multiseuil	X		

Le modèle d'orthogonalisation des vecteurs complémentés est non valide car il commet des erreurs en présence de certaines combinaisons linéaires des vecteurs enregistrés (section 4.2.3). Ce modèle a permis, néanmoins, de mettre en évidence le problème de corrélation croisée (section 4.2.4).

Le modèle d'orthogonalisation par recodage caractéristique de la composante semble valide (section 4.3.3) mais on ne peut garantir son fonctionnement (section 4.3.4) en grande dimension. Ce modèle nous a conduit à développer une nouvelle architecture permettant de garantir le bon fonctionnement du système.

Les trois autres modèles étudiés se sont avérés valides. Pour ces modèles, on a quantifié la taille de la CMM, les ressources mémoire et les ressources de calculs nécessaires.

5.2.2 Synthèse des résultats dimensionnement des CMM

Les résultats présentés dans cette section sont tous obtenus avec des vecteurs d'entrée de 128 bits de long : ceci permet de comparer les modèles plus facilement. Le Tableau 5.2 présente les résultats observés pour les simulations valides. Les résultats obtenus avec l'orthogonalisation d'une CMM pointant dans une RAM sont détaillés dans le Tableau 4.3 et le Tableau 4.4, ceux obtenus avec le modèle de randomisation des entrées sur une CMM binaire à seuil unique se trouvent dans le Tableau 4.8 et le Tableau 4.9, ceux obtenus avec le modèle de randomisation des entrées sur une CMM binaire multiseuil sont présentés dans le Tableau 4.11 et le Tableau 4.12. Afin de simplifier les notations, on appellera :

- Modèle I, le modèle d' orthogonalisation d'une CMM pointant dans une RAM.
- Modèle II, le modèle de randomisation des entrées sur CMM binaire à seuil unique
- Modèle III, le modèle de randomisation des entrées sur CMM binaire multiseuil

Le facteur de restriction est le facteur de réduction de l'espace de recherche que permet d'obtenir le prétraitement par la CMM.

Tableau 5.2 : Synthèse des ressources nécessaires pour les différents modèles valides

	Nombre de bits par enregistrement	Nombre d'additions	Nombre de multiplications	Facteur de restriction de l'espace de recherche
Modèle I	176	889 Nombres à virgule fixe de 25 bits	896 Nombres à virgule fixe de 25 bits	127
Modèle II	256	32512 Opérations d'incrémentation	32768 Opérations "ET" logiques	256
Modèle III	69	7665 Opérations d'incrémentation	7680 Opérations "ET" logiques	112

On constate que le modèle III est le plus économique pour le stockage des vecteurs. C'est le seul des trois modèles qui permet de stocker un vecteur de 128 bits dans un espace mémoire inférieur à 128 bits.

Il faut noter ensuite que le nombre d'opérations n'a pas la même signification pour le modèle I et pour les deux autres modèles. Dans le modèle I, les opérations sont faites avec des nombres à virgule fixe : une implantation matérielle de ce modèle nécessite des additionneurs et des multiplicateurs relativement complexes comparés aux opérations à effectuer dans les modèles II et III. Dans ces deux modèles, les opérations se font au niveau des bits : la multiplication est remplacée par un simple "ET" logique et l'addition est une simple opération d'incrémentation. Si l'on compare le modèle II et le modèle III, on constate que le modèle III nécessite quatre fois moins d'opérations.

Le modèle II est celui qui permet de restreindre le plus efficacement le domaine de recherche. Il est deux fois plus efficace que les modèles I et III dont les performances sont comparables dans ce domaine.

5.2.3 Les CMM binaires multiseuil, le meilleur des modèles étudiés

Les résultats comparés montrent que pour une taille de vecteurs d'entrée de 128 bits, le modèle III est le moins exigeant aussi bien en terme de ressources mémoires que de ressources de calcul. Toutefois, il n'est pas aussi performant que le modèle II pour restreindre l'espace de recherche et il faut deux entités du modèle III pour réaliser la même tâche de sélection que le modèle II. Toutefois, si l'on considère les ressources nécessaires pour implanter le modèle III, on constate qu'il est moins coûteux de faire fonctionner deux entités du modèle III en parallèle qu'une seule entité du modèle II. Ainsi, parmi les modèles étudiés le modèle III des CMM binaires multiseuil est celui qui, pour un niveau de performance identique, est le plus économique. Dans ce cas de figure, le traitement des vecteurs en lecture se fait ainsi :

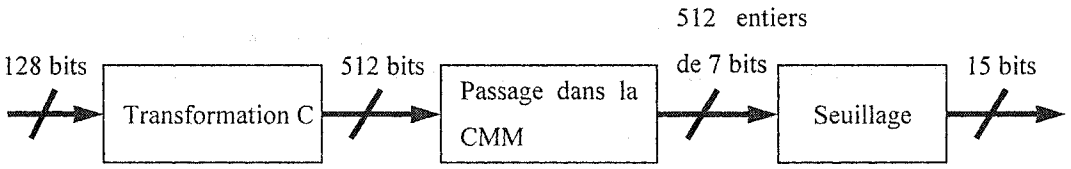


Figure 5.1 : Etapes de la lecture dans une CMM binaire multiseuil à 128 bits d'entrée

Afin de connaître le nombre de portes logiques nécessaires à cette architecture, il faut concevoir chacun de ces blocs : il faut, pour cela, envisager une implantation matérielle du système.

5.3 Éléments de comparaison des CAM « traditionnelles » et des CAM neuronales

Les travaux réalisés permettent de comparer les CAM « traditionnelles » et les CAM neuronales étudiées au niveau de leur fonctionnement. Les résultats obtenus donnent les

premiers éléments de comparaison en terme de performance et de coût entre ces deux architectures.

5.3.1 Comparaison des performances

La vitesse de traitement. S'il est indéniable que les CAM « traditionnelles » sont difficilement égalables en terme de vitesse de traitement, les CAM neuronales par randomisation des vecteurs d'entrée promettent de bonnes performances grâce à un prétraitement peu coûteux en ressources mémoire et grâce à une phase d'appariement très courte (sections 4.6.4.1 et 4.6.4.2).

La taille des vecteurs d'entrées. Les CAM « traditionnelles » sont mal adaptées aux grandes tailles de vecteurs d'entrée. Au contraire, l'augmentation de la taille des vecteurs est avantageuse dans les CAM neuronales utilisant des CMM binaires car une CMM possédant des vecteurs d'entrée plus grands offre un plus grand choix de seuils. Ainsi la CAM neuronale semble tout particulièrement adaptée au traitement des grands vecteurs d'entrée (section 4.7.2).

5.3.2 Comparaison des coûts

Coût de la mémoire de stockage. Le coût de la mémoire de stockage dans une CAM « traditionnelle » est directement relié au nombre de cellules mémoires dont dispose la CAM. Dans la CAM neuronale, la mémoire de stockage est une RAM, un type de mémoire très bon marché (section 4.6.1).

Ressources nécessaires. Si l'on calcule le nombre de portes logiques nécessaires pour concevoir une CAM élémentaire, on a, d'après la section 2.3.2, pour $n = 112$ et $s = 128$:

- 112*128 = 14336 porte "ET"
- 112*128/2 = 7168 porte "NON"

Ces valeurs sont du même ordre que les résultats obtenus avec des CMM binaires multiseuil (Tableau 4.12). Toutefois, pour pouvoir faire une comparaison précise, il faudrait disposer de résultats quantitatifs afin de connaître les ressources matérielles nécessaires pour l'intégralité du traitement (Figure 5.1).

Le partage entre traitement en série et traitement en parallèle des vecteurs de grande dimension dans une CAM neuronale. Dans une CAM neuronale, il est possible de choisir le degré de parallélisation des calculs : pour chaque bit de la sortie, il faut calculer une somme qui peut être calculée en plusieurs fois (section 4.5.1.3). Si l'on dispose de beaucoup de ressources matérielles, on peut effectuer le produit matriciel en une fois. Si l'on dispose de moins de ressources matérielles, on peut procéder à un calcul de la sortie de la CMM en plusieurs étapes. Cette capacité à choisir le partage entre traitement en série et traitement en parallèle rend la CAM neuronale beaucoup plus flexible que la CAM « traditionnelle » en terme de possibilités d'implantation.

5.4 Conclusion

L'étude des différentes implantations envisageables et la comparaison du modèle de CAM neuronales utilisant la randomisation des vecteurs avec les CAM « traditionnelles » montrent que les CAM neuronales offrent un grand choix d'implantations et qu'elles sont particulièrement adaptées aux vecteurs d'entrée de grande taille.

5.4.1 Un grand éventail de configurations

Avec les CAM neuronales, on peut accéder à différents niveaux de coûts et différents niveaux de performance suivant le niveau de parallélisation adopté et le choix d'implantation. Ainsi, en modifiant la Figure 2.3, on peut estimer que les CAM neuronales peuvent se situer sur une zone du plan coût-performance intermédiaire entre

les RAM et les CAM « traditionnelles » (Figure 5.2). Il serait utile de déterminer une fonction de coût analytique pour pouvoir déterminer une configuration optimale dans le plan vitesse d'exécution-coût matériel.

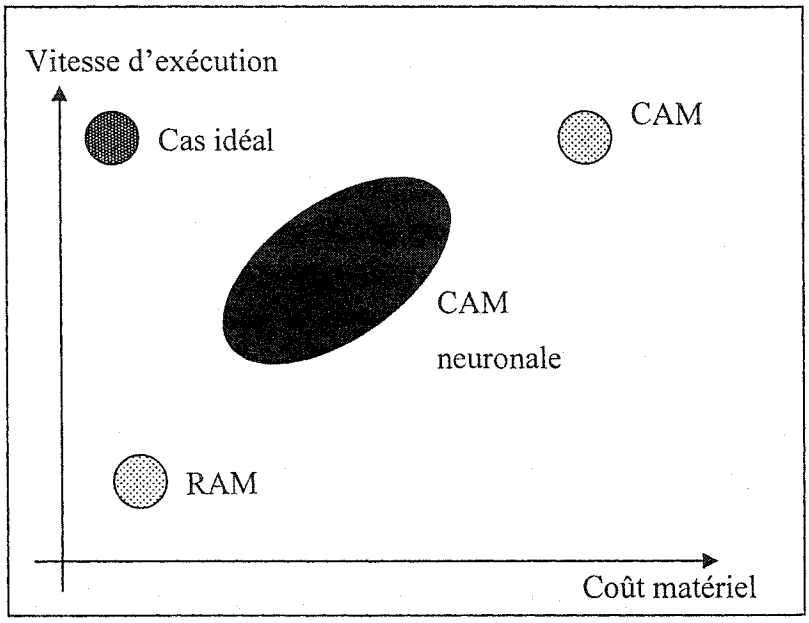


Figure 5.2 : Estimation de la localisation des CAM neuronales dans un plan coût-performance

Les simulations effectuées suggèrent que les CAM neuronales peuvent être utilisées dans des applications pour lesquelles on recherche une bonne vitesse de traitement sans vouloir investir dans une CAM « traditionnelle » dont le coût est trop élevé.

5.4.2 Un modèle adapté à de grands vecteurs d'entrée

L'autre avantage des CAM neuronales est qu'elles permettent de traiter des vecteurs d'entrée de grande taille avec un coût matériel moins important qu'une CAM « traditionnelle », en particulier avec une implantation logicielle ou une implantation mixte. En effet, de telles approches permettent d'effectuer le calcul de la sortie de la CMM en plusieurs fois. Théoriquement, on peut accéder à n'importe quelle taille de

vecteurs d'entrée en utilisant ce procédé : ceci revient en fait à calculer la sortie de la CMM en effectuant un produit matriciel par blocs.

5.4.3 Les paramètres non étudiées dans le modèle

Les travaux, s'ils ont permis de fournir des résultats quantitatifs pourraient être agrémentés de travaux supplémentaires. Ainsi, il serait particulièrement intéressant de tester les aspects suivants du modèle de CAM neuronales utilisant la randomisation des vecteurs.

Réduire la taille des entrées. Dans la méthode de randomisation des entrées, tous les bits du vecteur d'entrée sont randomisés : il doit être possible de ne randomiser qu'une partie de ces bits afin de réduire la taille du vecteur d'entrée de la CMM et d'ainsi améliorer le coût d'implantation.

Développer un algorithme non aléatoire. A partir des transformations B et C (section 4.5.1.4) on peut chercher à développer un algorithme maximisant l'espacement entre les vecteurs d'entrée à enregistrer dans la CMM. Néanmoins, il est possible qu'un tel algorithme soit difficile à mettre en œuvre.

Travailler avec de plus grands vecteurs. L'étude des CAM pour des vecteurs d'entrée plus grands permettrait d'avoir davantage de données quantitatives sur les CMM binaires : la limite que nous avions, était le temps de calcul nécessaire sous MATLAB pour les simulations. Une programmation du même modèle en C ou en C++ permettrait de travailler avec des tailles de vecteurs très grandes. Si on s'en réfère à la théorie, une augmentation de la taille des vecteurs d'entrée doit impliquer une plus grande capacité de stockage. Une telle étude est intéressante essentiellement en ce qui concerne la capacité de stockage et le coût du pré traitement par la CMM. Néanmoins, une taille de 128 bits est actuellement suffisante pour des règles de systèmes coupe-feu.

Si toutes les simulations possibles n'ont pas été réalisées, la méthodologie décrite dans les sections 4.5 et 4.6 permet de réaliser aisément des tests pour de grandes tailles de vecteurs d'entrée.

CHAPITRE 6
CONCLUSION

Notre recherche sur les CAM neuronales à base de CMM permet de déterminer, parmi les cinq modèles étudiés, quelles sont les architectures valides. Pour les modèles validés, l'étude des ressources nécessaires a servi à estimer quantitativement les ressources mémoires et les ressources de calcul requises. Les résultats de nos travaux permettent d'affirmer la possibilité d'utiliser des réseaux de neurones dans un modèle de CAM. Ces résultats encouragent à étudier les différentes implantations possibles de ces modèles de mémoire.

6.1 CONTRIBUTIONS APPORTÉES

Les simulations réalisées ont permis de déterminer le rôle que doit avoir un réseau de neurones dans une CAM neuronale et elles ont fourni, en outre, les informations permettant d'estimer l'intérêt des CAM neuronales du point de vue de la vitesse de traitement et du coût.

6.1.1 Inadaptation des CMM auto associatives

6.1.1.1 Difficultés d'implantation : le problème des combinaisons linéaires et des vecteurs proches au sens de Hamming

Si la littérature concernant les CMM présente systématiquement la possibilité de retrouver des enregistrements orthogonaux, elle n'étudie pas le comportement du système en présence de vecteurs non enregistrés. Dans nos travaux, nous avons fait

l'étude du comportement des CMM aussi bien pour des vecteurs enregistrés que pour des vecteurs non enregistrés.

D'après les résultats obtenus, il semble impossible d'utiliser les CMM comme des mémoires de stockage à part entière. En effet, un tel choix d'architecture pose plusieurs problèmes, nous avons découvert que :

- Des erreurs peuvent se produire comme cela a été le cas avec le premier modèle étudié. On a détecté des erreurs pour certains vecteurs qui avaient la particularité d'être systématiquement des combinaisons linéaires des vecteurs enregistrés : un tel comportement était difficilement prévisible a priori.
- Il est difficile de dimensionner les coefficients de la CMM : le deuxième modèle ne présente pas d'erreurs détectables mais il est très sensible à la distance de Hamming au plus proche enregistrement et la précision des coefficients de la CMM doit être d'autant plus grande que la taille des vecteurs d'entrée est grande. Ces résultats impliquent un besoin en ressources élevé si l'on veut traiter des vecteurs de grande taille.

6.1.1.2 Impossibilité de valider exhaustivement un tel système

Comme le comportement des CMM en présence des tous les vecteurs possibles n'a pas été étudiée, il en est de même pour la validation du comportement d'une CMM dans tous les cas de figures possibles.

Nous avons constaté l'impossibilité de valider un sytème tel une CMM auto associative. La validation du système pour une taille de vecteurs d'entrée donnée ne donne aucun renseignement sur la validité du modèle pour des tailles de vecteurs d'entrée plus grandes. De plus, la seule manière de certifier le bon fonctionnement d'un réseau de neurones serait de tester toutes les entrées possibles pour tous les états possibles du système.

Nous avons établi, étant donné l'immensité de l'espace des entrées d'une part et de l'espace des états possibles d'autre part, qu'une telle validation est irréalisable et ce, même pour des vecteurs d'entrée de taille très modeste : il est, par exemple, tout à fait impossible de valider un tel système de manière exhaustive avec des vecteurs d'entrée de 8 bits pour lesquels il faudrait effectuer plus d'un million de milliards de tests.

6.1.2 L'utilisation des réseaux de neurones pointant dans une RAM : une solution au problème de validation

Pour répondre au problème de validation dans les CAM neuronales, nous avons créé une architecture utilisant des réseaux de neurones afin de restreindre très significativement l'espace de recherche pour rélaiser une détection exacte des vecteurs. Utilisés ainsi, les réseaux de neurones permettent de réduire de plus de 99% le nombre de tests à effectuer.

6.1.3 Inadaptation de la méthode d'orthogonalisation exacte des entrées

Le modèle des CMM donne toute la base théorique permettant leur construction mais ne donne aucune information quand aux ressources nécessaires, notamment en ce qui concerne l'espace mémoire à allouer à chacun des coeffcients de la CMM. Notre recherche a permis de quantifier ces ressources. Ces résultats nous ont permis de conclure au coût élevé d'une telle architecture.

6.1.3.1 Des performances peu intéressantes

Pour le modèle d'orthogonalisation exacte des entrées, la taille des coefficients de la CMM a été évaluée à 32 bits pour une taille de vecteurs d'entrée de 120 bits. Comme le prétraitement par la CMM consiste en une multiplication matricielle avec des nombres de 32 bits, le temps nécessaire pour effectuer cette multiplication ne permet pas

d'obtenir de bonnes performances même si l'on dispose d'importantes ressources de calcul.

6.1.3.2 Un coût d'implantation élevé

En plus d'empêcher le système d'être performant, l'utilisation de coefficients de grande taille contraint à implanter des fonctions d'addition et de multiplication pour des nombres de 32 bits. Vu qu'il faut effectuer des centaines d'additions et de multiplications pour calculer la sortie de la CMM, il est préférable de disposer d'un grand nombre d'additionneurs et de multiplieurs si l'on désire un traitement rapide des données. Un tel choix d'implantation occasionne un coût matériel élevé et ne permet donc pas d'atteindre les objectifs fixés.

Avec un modèle utilisant l'orthogonalisation exacte des vecteurs d'entrée, on ne peut pas effectuer un prétraitement efficace : nous avons donc développé un autre modèle de CAM neuronale beaucoup moins exigeant en ressources de calcul.

6.1.4 Le modèle utilisant la randomisation des entrées : un modèle performant et peu coûteux

Pour le modèle utilisant randomisation, que nous avons créé -parmi les transformations des entrées, seule la transformation A a été trouvée dans la littérature-, nous avons quantifié les ressources mémoires et les ressources de calculs nécessaires.

Le modèle utilisant la randomisation des entrées a l'avantage d'utiliser une CMM binaire : ceci permet de bénéficier d'un prétraitement à la fois rapide et peu coûteux. Les simulations ont permis de valider ce modèle pour des vecteurs d'entrée de 128 bits. En outre, d'un point de vue théorique, l'augmentation de la taille des vecteurs d'entrée doit

favoriser le réglage des seuils sur les sorties de la CMM : ce modèle paraît donc adapté à des vecteurs d'entrée de grande taille.

Le principal avantage des CMM binaires est d'utiliser le "ET" logique comme multiplication en lecture, les additions consistent en de simples incrémentations. Ainsi le traitement des vecteurs peut être effectué très rapidement et le coût engendré par la CMM binaire est peu important du fait de la simplicité des fonctions à réaliser pour calculer le vecteur de sortie. Le modèle utilisant la randomisation des entrées bénéficie de nombreuses qualités qui devraient permettre de créer une CAM neuronale rapide pour un coût matériel relativement bas. Il est cependant nécessaire de tester différentes implantations de ce modèle pour confirmer ses bonnes performances.

6.2 VOIES À EXPLORER

L'étude effectuée sur les CAM neuronales à base de CMM montre que ces mémoires peuvent s'avérer très performantes. Ainsi, leur étude suggère des recherches dans de nouvelles directions :

- Il est possible de réaliser une étude des différentes implantations envisageables.
- Il semble possible de développer, à partir du modèle de CAM neuronale utilisant la randomisation, une CAM neuronale ternaire.
- Il serait intéressant d'étudier le modèle des SDM développé par Kanerva. C'est un autre modèle de CAM neuronale qui possède des propriétés proches de l'algorithme k-NN mais dont l'architecture permet de déterminer en quelques cycles les plus proches voisins d'un vecteur présenté en entrée. A l'instar des CMM, on doit pouvoir utiliser les SDM dans une CAM neuronale.

6.2.1 Différents supports d'implantation envisageables

Si les modèles de randomisation ont été validés, il reste à étudier leur implantation pour déterminer, de manière quantitative, le coût en ressources matérielles et les performances des CAM neuronales. On peut envisager trois possibilités : une implantation logicielle, une implantation matérielle ou une implantation mixte. Les résultats obtenus par Zhou [ZHO99] montrent que le développement de tels systèmes doit permettre une grande accélération du traitement.

6.2.1.1 Implantation logicielle

Tout d'abord une implantation logicielle en C ou en C++ peut s'avérer très efficace. En tirant parti des jeux d'instructions spécialisés, tels MMX, SSE ou SSE2 dans le cas des Pentium d'Intel, il est possible d'accélérer grandement la vitesse de calcul du vecteur de sortie de la CMM. En effet, ces jeux d'instructions offrent de bonnes optimisations pour les calculs matriciels. De plus, une telle approche ne nécessite aucune boucle conditionnelle durant le traitement par la CMM et seulement une ou deux boucles conditionnelles dans la phase d'appariement. Ce nombre restreint de branchements conditionnels permet un traitement plus efficace des données car ainsi, le processeur sait exactement les instructions qu'il doit appliquer.

De plus, une telle approche permettrait de traiter des règles de grande taille en séquençant les vecteurs d'entrée et en procédant à des produits matriciels par blocs au niveau de la CMM.

6.2.1.2 Implantation matérielle

Une implantation sur circuit intégré ou FPGA est envisageable et ceci permettrait une bonne parallélisation du traitement. Toutefois, le coût d'une implantation sur un FPGA

est plus élevé que celui des CAM « traditionnelles » et il existe déjà des IP permettant d'implanter des CAM « traditionnelles ». On peut néanmoins utiliser un FPGA comme plate-forme de tests pour déterminer les performances d'une CAM neuronale implantée matériellement.

6.2.1.3 Implantation mixte

Si on considère les deux approches précédentes, on réalise qu'une implantation mixte peut permettre de bénéficier d'une très bonne accélération matérielle tout en gardant une bonne flexibilité et un coût peu élevé. On pourrait, par exemple, utiliser un DSP pour accélérer le prétraitement par la CAM, une LUT pour transcrire les adresses de sorties de la CMM en adresses dans la RAM. De plus, la programmation des DSP permet de travailler avec de grandes tailles de vecteurs d'entrée en séquençant le traitement des vecteurs d'entrée.

6.2.2 Les CAM neuronales ternaires : une extension possible du modèle

Afin de faciliter la définition des règles de routage ou les règles d'un système coupe-feu, le développement de CAM ternaires serait pertinent. On pourrait utiliser les codes développés précédemment et les étendre au cas « don't care » (DC) qui permet de tester des ensembles de vecteurs. L'implantation d'un tel système nécessiterait une transformation des entrées plus complexe en raison de la présence de l'état DC.

6.2.3 Un autre modèle neuronal à étudier : les SDM de Kanerva

Le modèle des « Sparse Distributed Memories » de Kanerva est un modèle de mémoire distribué destiné, à l'origine, à évaluer le meilleur représentant possible d'un vecteur inconnu. Ce modèle, qui a l'avantage d'être facilement implantable en matériel, possède de grandes ressemblances avec les CMM binaires. Notamment, la principale hypothèse

de travail porte sur les vecteurs d'entrée : elle stipule que ces vecteurs doivent être « sparses » et de grande dimension. En raison des similitudes entre les SDM et les CMM, il serait intéressant d'étudier les possibilités d'intégrer des SDM dans une CAM neuronale.

Cette recherche propose des solutions alternatives aux CAM « traditionnelles ». Ce type de solution peut intéresser les fabricants de CAM : eux-mêmes investissent dans la recherche des modèles plus économiques que les CAM « traditionnelles » :

> «[...] CAMs traditionally have been greedy consumers of board space and power. Process shrinks have mitigated the problems, but it's still conceivable that OEMs might want lower-power alternatives [...] Even some CAM vendors are dabbling in new algorithms.»
> Craig Matsumoto, EE Times, mai 2002

De manière plus générale, le développement de méthodes performantes de recherches de données est utile partout où existent des bases de données et le développement d'alternatives aux CAM « traditionnelles » telles les CAM neuronales, plus économiques, va dans le sens d'une démocratisation de ce type très performant de mémoires.

BIBLIOGRAPHIE

[AUS87] AUSTIN, J. (1987). ADAM: A Distributed Associative Memory for Scene Analysis. Proceedings of First International Conference on Neural Networks, Ed. M Caudhill and C Butler, San Diego, Vol. IV, p. 285.

[AUS88] AUSTIN, J. (1988). Grey Scale N Tuple Processing. Pattern Recognition: 4th International Conference, Cambridge, UK. Lecture Notes in Computer Science, Vol. 301, Ed. Josef Kittler, Springer-Verlag, pp. 110-120

[AUS93] AUSTIN , J., BROWN, M., KELLY, I. et BUCKLE, S. (1993) ADAM Neural Networks for Parallel Vision. JFIT Technical Conference, pp. 173-180.

[AUS98] AUSTIN, J. (1998). RAM-Based Neural Networks. Progress in Neural Processing, Vol. 9. 200p.

[BAL99] BALLARD, D. H., (1999). An Introduction to Natural Computation, chap. 7. MIT Press. 336p.

[BLE59] BLEDSOE, W. W. et BROWNING, I. (1959) . Pattern Recognition and Reading by Machine. Proc. Eastern Joint Computer Conference, pp. 232-255.

[COO73] COOPER, L. N. (1973). A Possible Organization of Animal Memory and Learning. Proceedings of the Nobel Symposium on Collective Properties of Physical Systems. B. Lundquist and S. Lundquist editions, pp. 252-264.

[DELO99] DELORME, A., GAUTRAIS, J., VAN RULLEN, R. et THORPE, S. (1999). SpikeNET: A Simulator For Modeling Large Networks of Integrate and Fire Neurons. Neurocomputing, vol 26-27, pp. 989-996.

[DELG99] DELGADO-FRIAS, J. G., YU, A. ET NYATHI, J. (1999). A Dynamic Content Addressable Memory Using a 4-Transistor Cell. IEEE Third International Workshop on Design of Mixed-Mode Integrated Circuits and Applications, Puerto Vallarta, Mexico, pp. 110-113.

[DIT00] DITMAR, J.M. (2000). A Dynamically Reconfigurable FPGA-based Content Addressable Memory for IP Characterization. Mémoire de maîtrise, KTH-Royal Institute of Technology, Stockholm, Suède.

[FRE01] FREY, B.J., KRISTJANSSON, T., DENG, L. et ACERO A. (2001). Learning dynamic noise models from noisy speech for robust speech recognition. Advances in Neural Information Processing Systems 14: Proceedings of the 2001 Neural Information Processing Systems (NIPS) Conference. T. G. Dietterich and S. Becker and Z. Ghahramani (Editeurs). MIT Press, Cambridge, MA.

[GWE02] GWENNAP, L. (2002). Is it time for CAMs? EETimes, 2002 -06-03. http://www.eetimes.com/story/OEG20020603S0011.

[HAY98] HAYKIN, S. S. (1998). Neural Networks: A Comprehensive Foundation, Prentice Hall, 2nd edition. 842p.

[HOP82] HOPFIELD, J. J. (1982). Neural Networks and Physical Systems with Emergent Collective Computational Abilities. Proceedings of the National Academy of Sciences, USA, vol 79. pp. 2554-2558.

[IOA00] IOANNIDIS, S., KEROMYTIS, A. D., BELLOVIN, S. M. ET SMITH, J. M. (2000). Implementing a Distributed Firewall. ACM Conference on Computer and Communications Security. pp 190-199.

[KAN88] KANERVA, P. (1988). Sparse Distributed Memory. MIT Press. Cambridge, MA. 155p.

[KAN95] KANERVA, P. (1995). A Family of Binary Spatter Codes. International Conference on Artificial Neural Networks - ICANN 95, Vol. 1. Proceedings: F. Fogelman-Soulie and P. Gallineri éditeurs, EC2 & Cie, Paris, pp. 517-522.

[KAN96] KANERVA, P. (1996). Binary Spatter-Coding of Ordered K-tuples. International Conference on Artificial Neural Networks -- ICANN 96. Proceedings: C. von der Malsburg, W. von Seelen, J.C. Vorbruggen, and B. Sendhoff éditeurs, Springer, Berlin, pp. 869-873.

[KAN98] KANERVA, P. (1998) Encoding Structure in Boolean Space. International Conference on Artificial Neural Networks - ICANN 98. Proceedings: L Niklasson, M. Boden, and T. Ziemke éditeurs, Springer, Berlin, pp. 387-392.

[KOM01] KOMAR, B, BEEKELAAR, R. ET WETTERN, J. (2001). Firewalls for Dummies. Collection For Dummies. John Wiley & Sons. 384p.

[LIU01] LIU, H. (2001). Reducing Routing Table Size Using Ternary-CAM. Proc. Hot Interconnect 9, Stanford.

[LUC97] LUCAS, S. M. (1997). Face recognition with the continuous n-tuple classifier, British Machine Vision Conference.

[MAT02] MATSUMOTO, C. (2002). CAM vendors consider algorithmic alternatives. EETimes, 2002-05-20. http://www.eetimes.com/story/OEG20020520S0014

[MCA93] MCAULEY, A. J. et FRANCIS, P. (1993). Fast Routing Table Lookup Using CAMs, INFOCOM. pp. 1382-1391.

[MCC43] MCCULLOCH, W. S. et PITTS, W. (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics, vol. 5, pp 115-133.

[OPE99] OPEN SKY TECHNOLOGIES. (1999). MPbase, Where The Network Is The Database! Complete Documentation Set. Next Paradigm Systems Inc.

[PRE93] PRESS, W. H., FLANNERY, B. P., TEUKOLSKY, S. A. et VETTERLING, W. T. (1993). Numerical Recipes in C: The Art of Scientific Computing. Cambridge University Press, 2nd edition. p. 98.

[RUM86] RUMELHART, D. E., HINTON, G. E. et WILLIAMS, R. J. (1986). Learning Representations of Back-Propagation Errors. Nature (London), vol. 323, pp. 533-536.

[TUR97] TURNER, M. et AUSTIN, J. (1997). Matching Performance of Binary Correlation Matrix Memories, Neural Networks Elsevier Science, vol. 10, numéro 9. pp. 1637-1648.

[VIA02] VIAL, B. (2002). TCP/IP Pratique. Collection e-Poche. Micro Application. 478p.

[WIL01] WILSON, R. (2001). The Specialty Memory Rises. EETimes, 2001-10-01. http://www.eetimes.com/story/OEG20011001S0019

[ZHO99] ZHOU, P., AUSTIN, J. et KENNEDY, J. (1999). A High Performance k-NN Classifier Using a Binary Correlation Matrix Memory. Advances in Neural Information Processing Systems, Vol. 11, CA, MIT press. pp. 713-722.

[ZWI00] ZWICKY, E. D., COOPER, S., CHAPMAN, D. B. ET RUSSELL, D. (2000). Building Internet Firewalls (2nd Edition). O'Reilly & Associates. 869p.

ANNEXE I : PRINCIPE DE FONCTIONNEMENT D'UN FIREWALL

I.1 INTRODUCTION

Un système de protection tel qu'il en existe dans des réseaux de grande taille est à la fois très complexe et très difficile à maintenir. Cette annexe expose toutes les connaissances essentielles à la compréhension des systèmes coupe-feu ou « firewalls ». Le document présente tout d'abord les attaques auxquelles un réseau doit faire face, ce qui permet de comprendre quelles en sont les parties vulnérables. Sachant cela, il faut se doter des moyens de protection adéquats qui vont permettre de prévenir d'éventuelles attaques dirigées contre le réseau. Il s'agit ensuite d'organiser tous ces éléments de protection en un système de protection à part entière : ce qui constitue normalement le « firewall », est en fait une partie du système de sécurité même si on désigne souvent par « firewall » l'ensemble du système de sécurité. La dernière étape dans la conception du système de sécurité est la configuration dont le but est de protéger le réseau interne en pénalisant le moins possible les utilisateurs. Cette annexe est fortement inspirée des livres « Firewall for Dummies » [KOM01] et « Building Internet Firewalls, 2nd edition» [ZWI00].

I.2 LES ATTAQUES POSSIBLES

Les attaques contre un réseau peuvent être aussi bien internes qu'externes. Dans les 2 cas, les attaques ou menaces sont multiples. Le but du « firewall » est de minimiser les risques d'agression, voire de les éliminer.

I.2.1 Les menaces

Les différentes menaces peuvent être classées en quelques catégories dont voici une brève description :

- Les mots de passe facile à deviner : par exemple, pour des mots de passe de petite taille, une recherche exhaustive peut suffire.
- L'intrusion qui consiste à pénétrer dans le réseau interne à partir du réseau externe. Ce type d'attaques peut se faire par une des méthodes suivantes :
    - Les « portes arrière » (back door) : ce sont souvent des facilités laissées par le programmeur du logiciel pour développer son programme. Il arrive néanmoins que les programmeurs laissent volontairement ou involontairement des failles dans les systèmes qu'ils programment. De telles failles permettent, par exemple, l'exportation de programmes tels NetBus ou BackOrifice sur une des machines du réseau attaqué : ces programmes permettent de contrôler un ordinateur à distance sans être repérés.
    - Virus, « Worm » et Cheval de Troie : un virus est un programme qui va se répandre de fichier en fichier, un « worm » va, lui, se diffuser de machine en machine alors qu'un cheval de Troie va installer un programme dans la machine cible tout en ayant la forme d'un programme anodin (comme par exemple une carte d'anniversaire électronique).
- Le « Denial of Service » (DoS) qui consiste à rendre un service indisponible (exemple : l'attaque du serveur DNS pour connaître les adresses des différentes machines du réseau interne et donc accéder à la structure globale du réseau, le « flooding »)

- Le « Distributed Denial » of Service (DDoS) qui est un avatar du DoS dans lequel on prend le contrôle de nombreuses machines pour ensuite coordonner une attaque contre un serveur cible (exemple de l'attaque du serveur de Yahoo)
- « Man in the middle attack » : le principe de cette technique est d'intercepter une connexion légitime et ensuite soit d'écouter la communication (« eavesdropping ») ou de se faire passer pour l'interlocuteur légitime (« impersonation »).
- Les attaques internes : ce sont souvent les attaques les plus difficiles à éviter

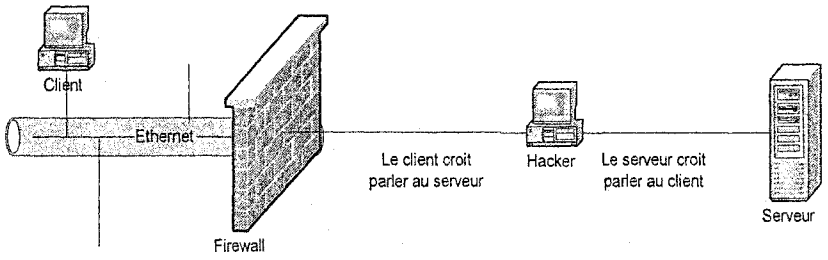


Figure I.1 :L'attaque par « Impersonation »

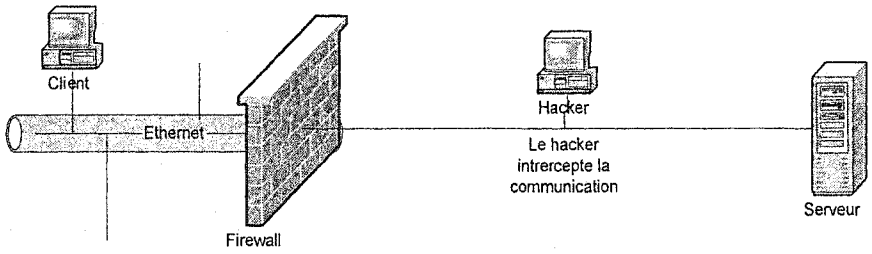


Figure I.2 : L'attaque par « Eavesdropping »

I.2.2 Que faut-il protéger ?

Dans un réseau, bon nombre de choses doivent être protégées :

- Les données confidentielles que les pirates vont essayer de récupérer
- Les données des clients comme les numéros de Cartes Bleues lors d'achats en ligne
- Les ressources matérielles que certains pirates vont essayer d'exploiter par exemple pour procéder à du DDoS (section I.2.1). De manière plus générale,l'utilisation des machines à distance qui est souvent employée par les pirates pour masquer leur piste.

I.2.3 Les types de protection

Pour se prémunir des attaques, il existe plusieurs types de protection qui sont décrits dans cette partie.

I.2.3.1 Filtrage de paquets

Le filtrage de paquets consiste en des vérifications dans l'en-tête des paquets (Annexe II). Il doivent être modifiés par les routeurs et les « firewalls » (au minimum, il faut décrémenter le TTL et recalculer le « checksum »). Le filtrage peut être « stateless » ou « stateful ». Un filtrage « stateful » va vérifier la cohérence d'une connexion en plus de vérifier la validité des paquets. En fait, le filtrage « stateful » permet de prévoir une partie des informations qui seront contenues dans les paquets à venir. Par exemple, lors d'une requête d'information HTTP, il faut ouvrir un port dont le numéro est supérieur à 1023 et qui est encore inutilisé. Dans un filtrage « stateless », on est obligé d'autoriser

tous les ports supérieurs à 1023 alors qu'un filtrage « stateful » permettra de n'ouvrir que les ports nécessaires. Le filtrage peut vérifier tous les renseignements suivants :

- L'IP source
- L'IP destination
- Le protocole (TCP, UDP, etc...)
- Le numéro de port pour un protocole TCP ou UDP : on parle ici du port de destination
- Le type de message ICMP (Internet Control Message Protocol) : certains type de message ICMP sont dangereux tel ICMP Redirect qui peut permettre du « eavesdropping »
- Les flags de fragmentation si les paquets IP sont recoupés en paquets plus petits.
- Les options IP qui sont souvent dangereuses

Le « IP spoofing » (utilisation d'une fausse adresse source) est un des dangers courants : une des vérifications que peut faire le « firewall » est de vérifier la validité des adresses sources au niveau des interfaces (une adresse interne lue au niveau de l'interface externe est probablement une attaque).

I.2.3.2 Le NAT'ing

Le « NAT'ing » permet de cacher la structure du réseau interne à toute personne extérieure au réseau local en exhibant des adresses externes différentes des adresses internes (le plus souvent, une seule adresse externe est disponible vu de l'extérieur). Les versions modernes du « NAT'ing » permettent à la fois la translation d'adresse et la translation de port (NAPT). Si le « NAT'ing » permet de masquer la structure du réseau, il comporte plusieurs inconvénients :

- Il est difficile d'identifier le « hacker » sans passer par l'étude du fichier journal ou « logfile » qui est le fichier qui enregistre toutes les données du trafic passant à travers le « firewall ».
- Pour certains protocoles, l'IP source se trouve à plusieurs endroits. Il faut effectuer les modifications à tous les endroits nécessaires sans quoi le paquet ne pourra être acheminé correctement.
- Un pirate peut tenter d'utiliser un port qui a été ouvert tant que le « mapping » du NAT entre le réseau externe et le réseau interne existe : il est donc préférable de fermer les ports une fois la connexion fermée.
- Il n'est pas possible de faire le « NAT'ing » si l'adresse source est présente dans la partie encryptée du message.

I.2.3.3 Le service mandataire (proxy)

Contrairement au filtrage de paquets, le service mandataire permet d'inspecter l'intérieur même des paquets, c'est-à-dire la partie de données de l'application. De plus, le service mandataire va reconstruire un paquet autorisé au lieu de simplement le rediriger. Ainsi, il y a 2 connexions séparées dans un « proxy » : une connexion interne et une connexion externe. Le « proxy » a les caractéristiques suivantes :

- Inspection complète du paquet
- Compréhension du protocole d'application
- Création d'un fichier de « log » très complet
- Non utilisation d'une véritable connexion interne/externe
- Aucun routage entre les interfaces réseaux (les paquets sont reconstruits)
- Possibilité d'inspecter des paquets à connexions multiples
- Possibilité de rechercher les informations dans le cache

On peut ainsi étendre la notion de règle abordée dans le filtrage de paquets :

- Examiner le contenu des données HTTP : par exemple interdire les vidéos
- Interdire certains noms de fichiers
- Vérifier l'intégrité des données comme, par exemple, détecter et interdire les fichiers contenant des virus
- Interdire les paquets contenant certains mots-clés.
- Inspection des e-mails sur le protocole SMTP : interdire par exemple certains messages attachés, certaines adresses de destination
- FTP get/put, SNMP get/set : par exemple pour autoriser la lecture mais interdire l'écriture

De plus, indépendamment du firewall, il est possible de bloquer un nom de site ou une adresse IP, de changer de police suivant le moment de la journée, d'appliquer des règles à un utilisateur ou a un groupe, gérer les quotas du FTP.

I.2.3.4 Surveiller et tracer les connexions (Monitoring and logging)

Le « firewall » garde une trace des différentes connexions sous forme d'un fichier. En plus de faire respecter des règles, plusieurs raisons encouragent à garder une trace « écrite » des connexions :

- Donner des renseignements sur les performances et l'utilisation du « firewall »
- Détecter les éventuelles intrusions (section I.2.1)
- Découvrir les méthodes d'attaques des pirates
- Avoir une preuve légale des intrusions

I.2.3.5 Détection d'intrusion

Grâce au « monitoring », il est possible de détecter des tentatives d'intrusion. En outre, cette opération n'interrompt pas le trafic. Le principe de la détection est d'identifier des « patterns » ou les étapes caractéristiques d'attaques dans le trafic. Typiquement, un tel système connaît une liste de signatures caractéristiques des tentatives d'intrusion telles :

- Le transfert de zone DNS
- Le scan d'adresse : pour savoir quelles adresses donnent une réponse
- Le scan de port : pour savoir quels ports sont ouverts
- Envoi de paquets corrompus

La réponse apportée à de telles tentatives pourra être :

- Enregistrer le problème
- Déclencher une alarme
- Modifier la configuration du « firewall » (qui peut être automatique)
- Lancer une contre-attaque

Les 2 dernières options comportent de nombreux risques et sont par conséquent peu recommandables.

I.3 STRUCTURE D'UN FIREWALL

Il existe plusieurs architectures communes de « firewalls ». Les plus rudimentaires ne possèdent qu'un seul point de protection, les plus évoluées en possèdent deux voire plusieurs.

I.3.1 Dual-Homed firewall

Ce type de « firewall » est constitué d'une interface externe reliée à l'internet et d'une interface interne reliée au réseau local. Cette architecture est simple à configurer et permet le « NAT'ing » mais elle ne comporte qu'un point de protection, ce qui implique que si le « firewall » est mis à défaut, tout le réseau est accessible à un « hacker ».

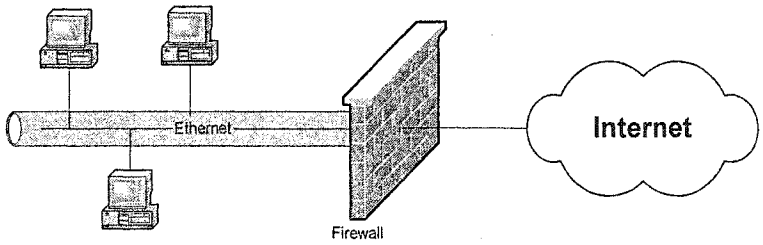


Figure I.3 : Exemple d'un "Dual Homed Firewall"

I.3.2 L'hôte caché ou screened host

On peut également envisager l'utilisation d'un hôte caché qui est dédié à un service particulier. On peut choisir ou non d'attribuer une sécurité accrue à cet hôte : si cet hôte bénéficie d'une protection particulièrement élevée, on l'appelle un bastion (« bastion host »). Dans le cas de certaines connexions sécurisées (tel L2TP[i]), il est nécessaire de contourner le « NAT'ing » et donc d'utiliser un hôte caché. La plupart du temps, si l'on dispose d'un réseau d'entreprise, il est nécessaire de posséder un serveur DNS. Pour des raisons de sécurité, il est préférable que seul le serveur DNS soit autorisé à envoyer des requêtes DNS vers l'internet. Ce serveur DNS, comme de nombreux serveurs, est

---

[i] L2TP est un protocole de tunneling qui utilise IPsec

souvent situé dans une partie spéciale du réseau appelée zone démilitarisée ou «DeMilitarized Zone » (DMZ).

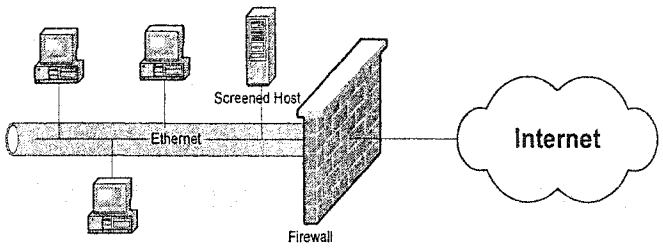


Figure I.4 : Screened Host

I.3.3 La zone démilitarisée (DMZ)

Le nom de zone démilitarisée vient du domaine militaire : en effet, il existe de nombreuses analogies entre une DMZ militaire et la DMZ d'un réseau informatique. Il faut bien être conscient que cette zone est un zone pacifiée et non pacifique : cela se traduit par des mesures de protection strictes en son sein. Les particularités de la DMZ d'un réseau, à l'instar d'une DMZ militaire, sont :

- Tout le trafic qui rentre et qui sort de la DMZ est inspecté
- Les ressources de la DMZ sont très fréquemment inspectées pour s'assurer que la sécurité n'est pas compromise
- La DMZ sert de frontière, de zone tampon pour protéger le réseau interne

Parmi les configurations utilisant une DMZ, on peut citer les « Three Pronged Firewalls » et les « Multiple Firewall DMZ ».

I.3.3.1 Three Pronged Firewall

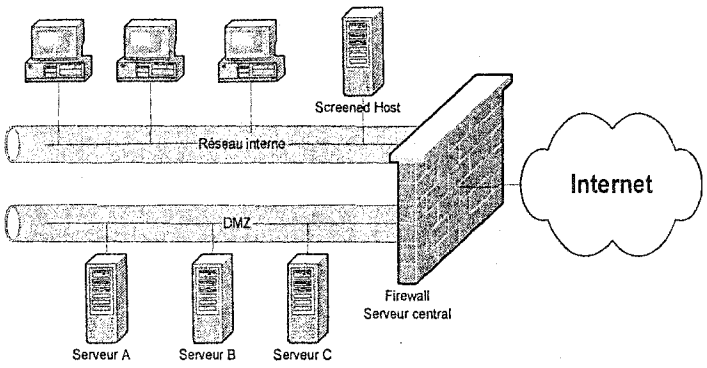


Figure I.5 : Three Pronged DMZ

Un tel « firewall » comporte 3 interfaces : un reliée au réseau interne, une reliée à l'internet et une dernière reliée à la DMZ. Une telle configuration permet de sécuriser les serveurs dédiés du côté de la DMZ et d'avoir le réseau local du côté de l'interface interne. L'avantage de cette configuration est qu'elle centralise les interfaces sur une seule machine ce qui facilite la mise en place et la maintenance du « firewall » tant au niveau logiciel qu'au niveau matériel. En contrepartie, la sécurité est moins grande que dans une configuration à « firewalls » multiples car si le « hacker » parvient a contrôler le serveur central, il peut accéder aussi bien aux machines situées dans la DMZ qu'aux machines du réseau local. De plus la charge de travail demandée (connexions internet/intranet, connexions internet/DMZ, connexions DMZ/intranet) peut vite créer un goulot d'étranglement au niveau du serveur central.

I.3.3.2 Multiple Firewall DMZ

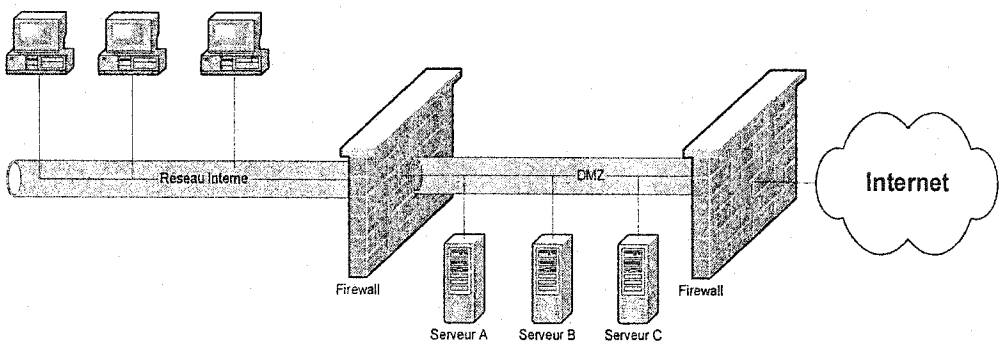


Figure I.6 : Multiple Firewall DMZ

La particularité de cette architecture est qu'elle fournit 2 ou plusieurs points de protection. Dans la cas d'une architecture à 2 « firewalls », le fait de distribuer l'inspection des paquets sur les 2 « firewalls » accroît grandement la sécurité. Toutefois, cette architecture comporte également plusieurs inconvénients :

- Le coût d'une telle architecture est élevé
- Il est nécessaire au responsable de la sécurité d'avoir très bonne connaissance des « firewalls »
- Le système de protection est plus long à administrer car il comporte 2 « listings » de filtrage
- Il est nécessaire de disposer d'utiliser plusieurs outils pour assurer un bon fonctionnement du système

La complexité du système devient encore plus grande si plus de 2 « firewalls » sont déployés. Dans une telle configuration, la meilleure stratégie d'administration est souvent de développer les connexions de l'interne vers l'externe, c'est-à-dire de traiter

tout d'abord les connexions entre réseau interne et DMZ puis de se concentrer sur la connexion entre la DMZ et internet.

I.3.3.3 Configuration hybride

Il est parfois nécessaire d'avoir à la fois une DMZ à adresses publiques et une DMZ à adresses privées, il est alors envisageable d'utiliser une architecture à 3 « firewalls », comportant une DMZ à adresses publiques et une DMZ à adresses privées. Il est à noter que l'on peut multiplier le nombre de « firewalls » à traverser, ce qui devrait résulter en une sécurité accrue. Toutefois, la complexification du système peut desservir son efficacité car il devient alors très difficile à administrer. De plus, il y a dans ce cas plus de règles à tester et il s'ensuit généralement une perte de débit. Ainsi, il est indispensable de simplifier le plus possible le système et donc trouver le meilleur compromis sécurité/simplicité aussi bien pour ne pas trop pénaliser le débit que pour diminuer les coûts du système. Par exemple, pour le tunneling, le logiciel *Firewall-1* de *CheckPoint* utilise le « firewall » lui-même comme terminal, ce qui permet une très grande simplification du système. Les méthodes de protection étant connues, il faut ensuite les regrouper en un système de protection complet et efficace.

I.4 CONCEVOIR SON PROPRE FIREWALL

La première étape dans la conception d'un « firewall » est de déterminer ce qui est autorisé et ce qui n'est pas autorisé. Dans un deuxième temps, il faut se donner les moyens de faire respecter les règles édictées en décrivant la politique de sécurité. Il faut ensuite définir les règles du « firewall » protocole par protocole.

I.4.1 Etablir une police d'utilisation

Afin de mettre en place une police d'utilisation du réseau qui corresponde aux attentes de l'entreprise, il est utile de se poser les questions suivantes :

- Questions sur ce qui est autorisé :
  - Quels services doivent être disponibles ? (quels programmes et quels protocoles peuvent utiliser les employés ?)
  - Qui peut accéder à internet ?
  - Quelles données peuvent être téléchargées à partir d'internet ?
  - A qui appartiennent les données ?
  - Quelle est la taille maximale autorisée pour les e-mails ?
- Questions sur ce qui n'est pas autorisé :
  - Quels types d'e-mails ou de fichiers attachés ne sont pas autorisés
  - Définir quelles actions peuvent être mises en œuvre sur internet (par exemple, certaines compagnies refusent que leurs employés expriment des opinions qui ternissent leur image de marque…)
  - Quels types de fichiers ne sont pas accessibles
  - Quels contenus web ne sont pas accessibles
  - Quelles connexions internet ne sont pas autorisées (exemple : tentative de connexion à un fournisseur d'accès internet à partir du réseau interne)
  - Quels protocoles, quelles applications ne sont pas autorisés

Il est aussi nécessaire de décider quelles sanctions appliquer en cas de non respect de la police d'utilisation du réseau.

I.4.2 Mettre en place la politique de sécurité

Afin d'établir une politique de sécurité efficace, les conseils suivant sont utiles à suivre :

- Faire de la mise en place de la politique de sécurité un travail d'équipe afin d'aboutir à un consensus quant à l'utilisation du réseau
- Identifier les ressources nécessitant une protection tant au niveau logiciel que matériel, tant au niveau des programmes que des données.
- Déterminer les probabilités correspondant aux différents risques encourus
- Créer un plan de réduction des risques pour chacune des menaces
- Déterminer les tâches des serveurs situés dans la DMZ du point de vue de la sécurité.
- Développer les scénarios à appliquer en cas de découverte de brèches dans le système de protection (notamment quelles mesures prendre tant que la brèche n'est pas colmatée).

Une fois la politique de sécurité définie, il ne reste qu'à générer les règles correspondantes du système de protection.

I.4.3 Etablir les règles

Les règles à établir sont plus ou moins simples à implémenter. A partir de 2 exemples, il est possible d'apprécier la méthodologie à employer.

I.4.3.1 Règles pour un protocole simple

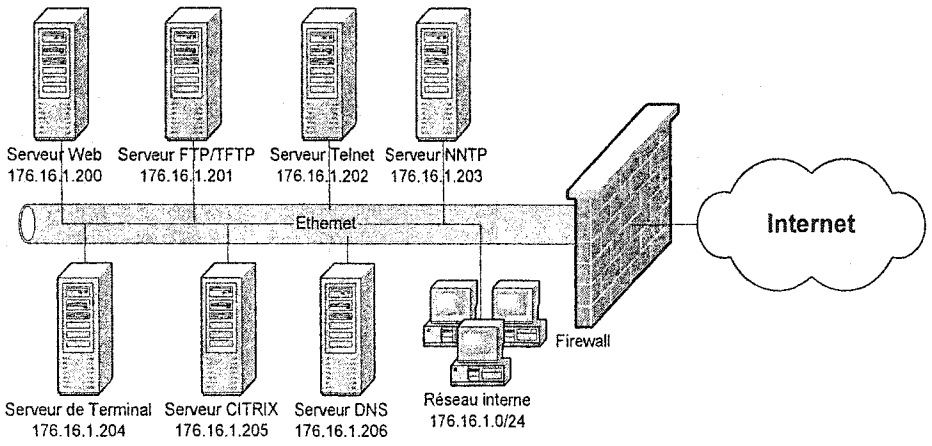


Figure I.7 : Exemple pour les protocoles simples

Services courants. Avec un tel réseau, les services tels l'accès au serveur web (port 80 et 443), au serveur DNS (port 53) se font via un port connu : toute requête sur ce port et appelant un service valide sera donc transférée sur le serveur correspondant. Pour ce qui est du FTP, 2 ports sont utilisés au niveau du serveur : le port 21 qui est le port de contrôle par lequel les commandes FTP sont transmises et le port de données qui est habituellement le port 20. Il existe cependant un mode FTP passif dans lequel le client et le serveur FTP négocient le port de données du serveur : c'est un mode considéré dangereux car il permet un moins bon contrôle des connexions. Le port de données du client sera, quant à lui, un port aléatoire de numéro supérieur à 1023.

Il est à noter qu'en FTP, la transmission du mot de passe est non cryptée ce qui peut constituer un grand danger pour la sécurité.

Thin Client Solutions. Il est parfois utile d'utiliser des solutions dans lesquelles le client ne dispose pas d'une grande capacité de calcul. Dans ce cas, c'est le serveur qui dispose

de la puissance de calcul : on peut utiliser pour cela soit un serveur Citrix, soit les services du terminal windows.

Serveur de news. Le serveur de News NNTP (ports 119 et 563) permet de lire les messages d'un newsgroup, etc. Le serveur de news est organisé un peu comme un serveur DNS en classant les groupes suivant leur domaine de discussion.

Telnet. Telnet (port 23) permet d'administrer un serveur à distance mais il est dangereux de l'utiliser car l'authentification n'est pas cryptée et il suffit d'intercepter la connexion pour connaître le mot de passe du client.

I.4.3.2 Règles pour des protocoles avancés

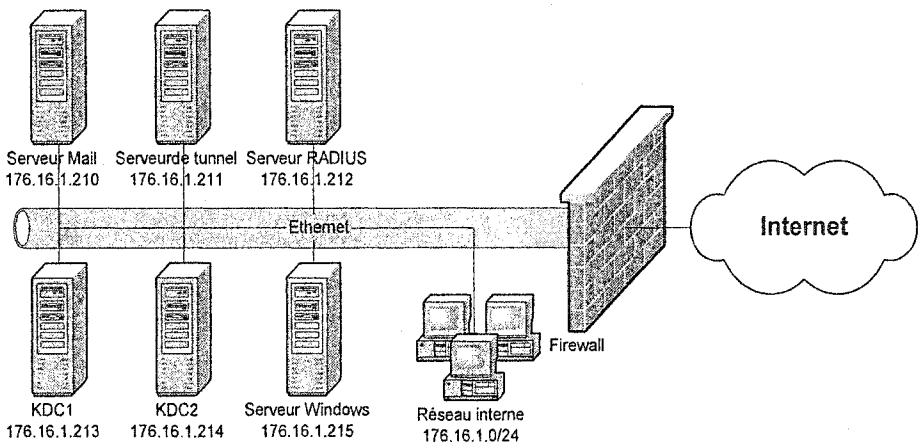


Figure I.8 : Exemple pour les protocoles avancés

Protocoles de courrier électronique. Il existe plusieurs protocoles pour le courrier électronique :

- POP3 (port 110) : c'est un protocole encore beaucoup utilisé mais qui a pour défaut d'accepter les courrier entrant et le courrier sortant.
- IMAP4 (port 143) : ce protocole permet d'accéder à sa messagerie sur un serveur. Il dispose de plusieurs dossiers en plus de la boîte de réception
- SMTP (port 25) : ce protocole permet l'envoi d'e-mails
- LDAP (port 389) : permet de trouver l'adresse e-mail d'une personne à partir d'un alias
- HTTP (port 80) : la consultation se fait en utilisant le serveur web qui lui-même envoie de requêtes à un serveur POP3.

Il faut noter qu'il existe des versions sécurisées de tous ces protocoles :

- Kerberos : nom anglais du chien mythologique Cerbère, le serveur Kerberos permet la restriction d'accès aux données aux seules personnes autorisées. Ce serveur permet le contrôle d'accès aux données aussi bien aux clients internes qu'aux clients connectés via internet.
- RADIUS : ce serveur permet des connexions par ligne téléphonique. Pour plus de détails, on peut consulter le livre « Firewalls for Dummies ».

ANNEXE II : DESCRIPTION DES EN-TÊTES DE PAQUETS IP ET TCP

Les protocoles IP (Internet Protocol) et TCP (Transport Control Protocol) sont deux protocoles couramment utilisés pour le transport de données à travers l'internet. Cette annexe présente une description de ces 2 protocoles très largement inspirée de « TCP/IP Pratique » [VIA02].

II.1 EN-TÊTE D'UN PAQUET IP

Le protocole IP a pour objectif de faciliter le transport des données de l'envoyeur au destinataire. L'en-tête d'un paquet IP (Figure II.1) contient un certain nombre d'informations qui doivent permettre le bon acheminement du paquet. L'en-tête IP contient les informations suivantes :

- Version : précise la version du format de l'en-tête.
- IHL (Internet Header Length) ou longueur de l'en-tête IP en mots de 32 bits avec une valeur minimale de 5 (sans options) et une valeur maximale de 15.
- Type de service : Ce champ contient des informations essentiellement destinées aux équipements d'interconnexion.
- Longueur totale : codée sur 16 bits, représente la longueur totale du datagramme mesurée en octets incluant l'en-tête IP et les données à transporter.
- Identificateur : Codé sur deux octets, ce champ constitue une identification utilisée pour reconstituer les différents fragments d'un même message . Tous les datagrammes du message porteront le même numéro d'identification.

- Flags : Ce champ occupe 3 bits et gère la fragmentation des paquets. Les valeurs de ces trois bits peuvent être :
    - 000 dernier fragment, ou seul fragment
    - 001 autorise la fragmentation, ce n'est pas le dernier fragment.
    - 010 la fragmentation n'est pas autorisée.
- Offset du fragment : indique la position fragment, comptée en unités de 8 octets par rapport au début des données du datagramme initial. Si le datagramme est complet ou si c'est le premier fragment, ce champ est à 0.
- TTL : indique la durée de vie maximale du datagramme au travers du réseau. A chaque traversée de passerelle, ce champ est décrémenté. Arrivé à 0, le datagramme est supprimé.
- Protocole : identifie le protocole de la couche supérieure
- Checksum d'en-tête : fait le checksum sur l'en-tête IP uniquement

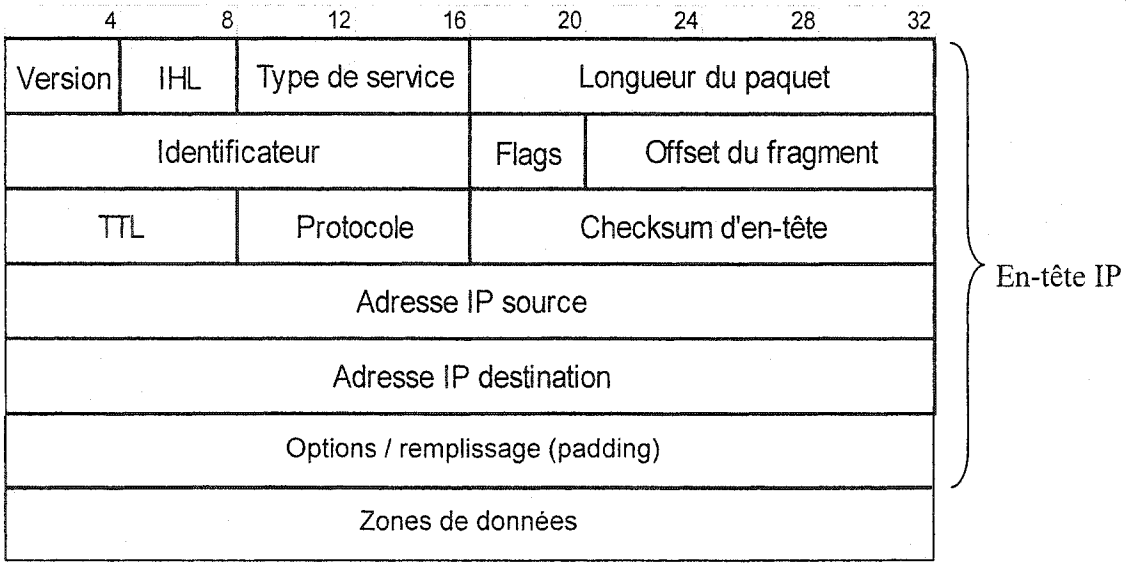


Figure II.1 : Datagramme d'un en-tête de paquet IP

II.2 EN-TÊTE D'UN PAQUET TCP

Le protocole TCP a pour objectif de vérifier l'intégrité des données transmises. L'en-tête d'un paquet TCP (Figure II.2) contient un certain nombre d'informations qui doivent permettre d'assurer que toutes les données transmises ont bien été reçues.

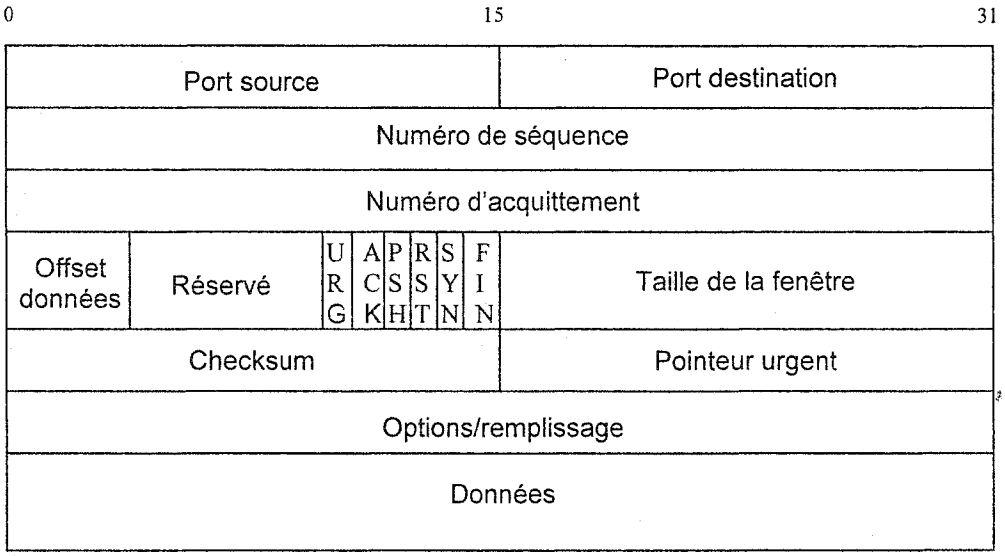


Figure II.2 : Datagramme d'un en-tête de paquet TCP

L'en-tête TCP contient les informations suivantes :

- Port source : numéro du port de la source (de 0 à 65535).
- Port destination : numéro de port du destinataire (de 0 à 65535).
- Numéro de séquence : le numéro du premier octet de données par rapport au début de la transmission (sauf si SYN est marqué). Si SYN est marqué, le numéro de séquence est le numéro de séquence initial (ISN) et le premier octet à pour numéro ISN+1.

- Numéro d'acquittement : si ACK est marqué ce champ contient le numéro de séquence du prochain octet que le récepteur s'attend à recevoir. Une fois la connexion établie, ce champ est toujours renseigné.
- Offset de données : la taille de l'en-tête TCP en nombre de mots de 32 bits. Il indique là où commencent les données. L'en-tête TCP, a une taille correspondant à un nombre entier de mots de 32 bits.
- Réservé : réservés pour usage futur. Doivent être à 0.
- Bits de contrôle :
    - URG: Pointeur de données urgentes significatif
    - ACK: Accusé de réception significatif
    - PSH: Fonction Push
    - RST: Réinitialisation de la connexion
    - SYN: Synchronisation des numéros de séquence
    - FIN: Fin de transmission
- Taille de la fenêtre : le nombre d'octets à partir de la position marquée dans l'accusé de réception que le récepteur est capable de recevoir.
- Checksum : voir la littérature pour connaître la méthode de calcul du checksum
- Pointeur de données urgentes : communique la position d'une donnée urgente en donnant son décalage par rapport au numéro de séquence. Le pointeur doit pointer sur l'octet suivant la donnée urgente. Ce champ n'est interprété que lorsque URG est marqué.
- Options : variable Les champs d'option peuvent occuper un espace de taille variable à la fin de l'en-tête TCP. Ils formeront toujours un multiple de 8 bits. Toutes les options sont prises en compte par le Checksum. Un paramètre d'option commence toujours sur un nouvel octet. Il est défini deux formats types pour les options:

- Cas 1: Option mono-octet.
- Cas 2: Octet de type d'option, octet de longueur d'option, octets de valeurs d'option. La longueur d'option prend en compte l'octet de type, l'octet de longueur lui-même et tous les octets de valeur et est exprimée en octets. Notez que la liste d'option peut être plus courte que ce que l'offset de données pourrait le faire supposer. Un octet de remplissage (padding) devra être dans ce cas rajouté après le code de fin d'options. Cet octet est nécessairement à 0. TCP doit implémenter toutes les options.

ANNEXE III : DÉNOMBREMENT DES ENREGISTREMENTS POSSIBLES DANS UNE CMM

Le nombre d'états possibles[ii] d'une mémoire pouvant enregistrer jusqu'à n mots de k bits est $\sum_{i=1}^{n}\binom{2^k}{i}=\sum_{i=1}^{n}\frac{(2^k)!}{i!(2^k-i)!}$. En effet la mémoire peut contenir 1, 2, ….n-1 ou n mots. Et comme il existe 2^k mots possibles, il y a $\binom{2^k}{i}$ façons de choisir i vecteurs différents dans l'ensemble des vecteurs de k bits.

Dans le cas de la CMM étudiées, on peut enregistrer jusqu'à (k-1) vecteurs : il y a donc $\sum_{i=1}^{k-1}\binom{2^k}{i}=\sum_{i=1}^{k-1}\frac{(2^k)!}{i!(2^k-i)!}$ états possibles de la mémoire. La Figure III.1de l'évolution du nombre de possibilités en fonction de la taille des vecteurs d'entrée montre que le nombre d'enregistrement devient vite considérable : la validation du modèle ne peut se faire que sur une petite partie des échantillons possibles. Les calculs sous MATLAB donnent pour valeur '*Inf*' pour une taille de vecteurs d'entrée supérieure à 34 bits, le graphe s'arrête à cet endroit. On constate que pour une taille d'entrée de 8 bits, il existe plus d'un million de milliards d'états possibles, ce qui empêche, pour cette taille déjà, toute étude exhaustive du comportement du système en fonction de son état.

[ii] On suppose qu'un état est défini entièrement par les vecteurs qui sont enregistrés et que, par conséquent, l'ordre d'enregistrement des vecteurs n'intervient pas.

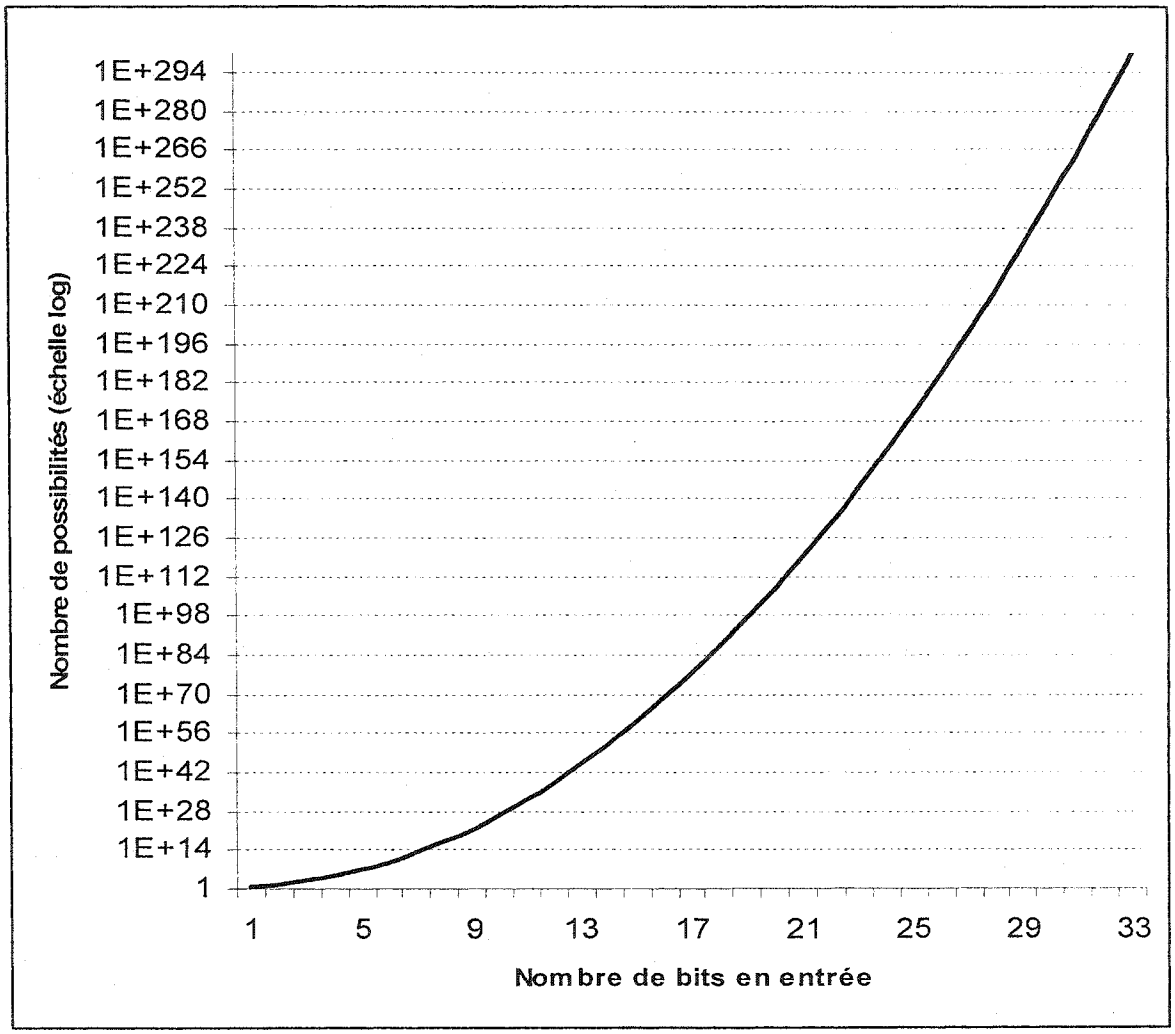


Figure III.1 : Evolution du nombre d'états possibles en fonction de la taille de l'entrée

Tableau III.1 : Nombre d'enregistrements possibles suivant la taille de l'entrée

Nombre de bits en entrée	Nombre d'enregistrements possibles
2	4
3	36
4	696
5	41448
6	8303632
7	5,699E+09
8	1,354E+13
9	1,126E+17
10	3,323E+21
11	3,517E+26
12	1,349E+32
13	1,894E+38
14	9,805E+44
15	1,883E+52
16	1,349E+60
17	3,624E+68
18	3,664E+77
19	1,399E+87
20	2,024E+97
21	1,113E+108
22	2,331E+119
23	1,864E+131

Tableau III.1 : Nombre d'enregistrements possibles suivant la taille de l'entrée (suite)

Nombre de bits en entrée	Nombre d'enregistrements possibles
24	5,702E+143
25	6,688E+156
26	3,012E+170
27	5,217E+184
28	3,481E+199
29	8,958E+214
30	8,903E+230
31	3,422E+247
32	5,090E+264
33	2,934E+282
34	6,561E+300

ANNEXE IV : LES SPARSE DISTRIBUTED MEMORIES

IV.1 INTRODUCTION

Le modèle des « Sparse Distributed Memories » ou SDM a été développé par Pentti Kanerva [KAN88]. Les SDM, à l'instar des CMM, constituent un modèle de mémoire distribuée à base de réseaux de neurones. Dans le modèle des SDM, on suppose que l'espace des entrées est de très grande dimension, c'est-à-dire de l'ordre de 100 bits au minimum. Cette annexe présente une courte description des processus de création des CMM et la méthode de récupération des données. La description des SDM faite dans cette annexe est fortement inspirée du livre de Kanerva.

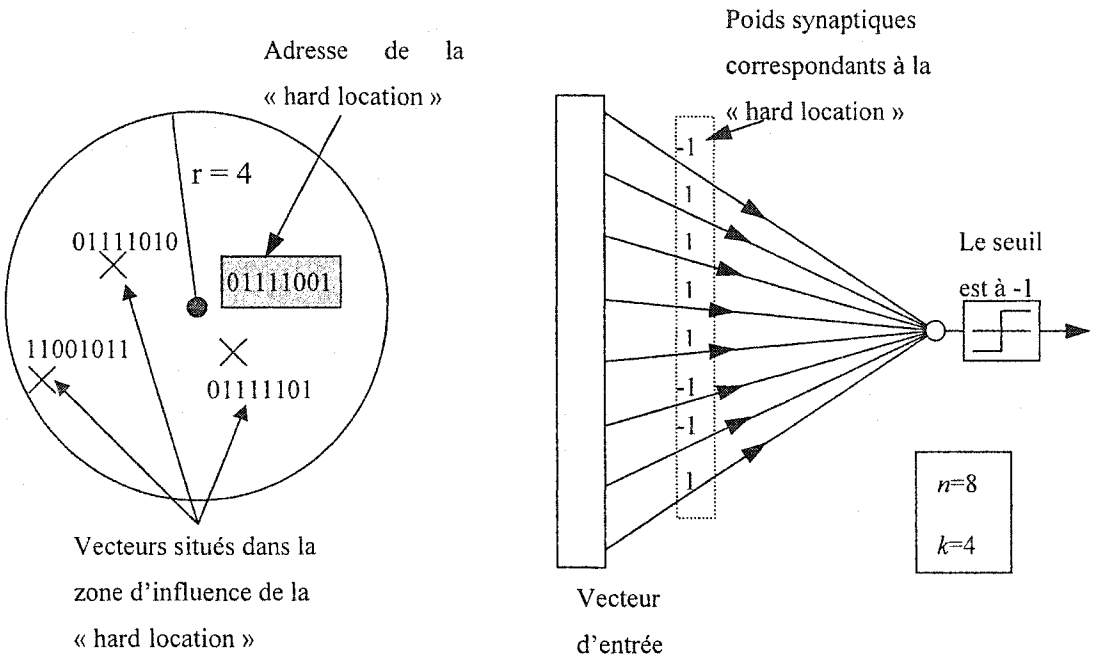
IV.2 CRÉATION DE LA SDM

Soit $E=\{0,1\}^n$, l'espace des vecteur d'entrée de la SDM. Le processus de création de la SDM nécessite la sélection de points de E appelés « hard locations ». Ces « hard locations » constituent les lieux d'enregistrement des données. Chaque « hard location » possède une zone de mémoire constituée d'un mot binaire.

IV.2.1 Les « hard locations »

Dans le modèle proposé par Kanerva, les « hard locations » sont situées en certains points de l'espace des données. Elles sont caractérisée par une zone d'influence définie

avec une distance de Hamming[iii] : Ces « hard locations » peuvent être représentée comme des neurones (Figure IV.1) et ont l'avantage d'être facilement implémentables sur un support matériel.



a) Représentation de la « hard location » dans E b) Représentation neuronale de la « hard location »

Figure IV.1 : Représentations d'une « hard location »

Notons w_{ij} les poids synaptiques de la $i^{ème}$ « hard location », alors on a :

- $w_{ij} = 1$ si le bit à mémoriser est 1
- $w_{ij} = -1$ si le bit à mémoriser est 0

[iii] La distance de Hamming entre 2 vecteurs est le nombre de bits dont ils diffèrent

La sortie v non seuillée du neurone est déterminée par :

$$v = \sum_{j=1}^{n} w_{ij} \qquad \text{(IV.1)}$$

Cette sortie est ensuite seuillée : le seuil est fonction de la zone d'influence de la « hard location ». Si on veut que la « hard location » traite tous les vecteurs qui lui sont distants d'au plus k bits, alors le seuil est réglé à (n-2k-1). Lorsqu'un vecteur d'entrée se trouve à k ou moins de k bits d'une « hard location », on dit qu'il active cette « hard location ». La taille de la zone d'influence, déterminée par k, est identique pour chaque « hard location ». On note W_i le mot binaire attaché à la $i^{ème}$ « hard location ». On suppose que tous les W_i ont une taille de n_w bits. On note W_{ij} le $j^{ème}$ bit de la $i^{ème}$ « hard location ».

Comme le modèle des SDM s'appuie sur des considérations probabilistes, Kanerva place les « hard locations » sur l'espace des données de manière aléatoire et avec une distribution uniforme.

IV.2.2 L'écriture dans les « hard locations »

Dans la phase d'enregistrement, on stocke les données D relatives au vecteur d'entrée dans chacune des « hard locations » qu'il active. On suppose que ces données ont la même taille que les W_i. Notons D_j le $j^{ème}$ bit de D, le processus d'écriture dans la « hard location consiste en l'opération suivante :

$$\forall j \in [1, n_w] \; W_{ij} = W_{ij} + D_j \qquad \text{(IV.2)}$$

La Figure IV.2 illustre ce processus d'enregistrement :

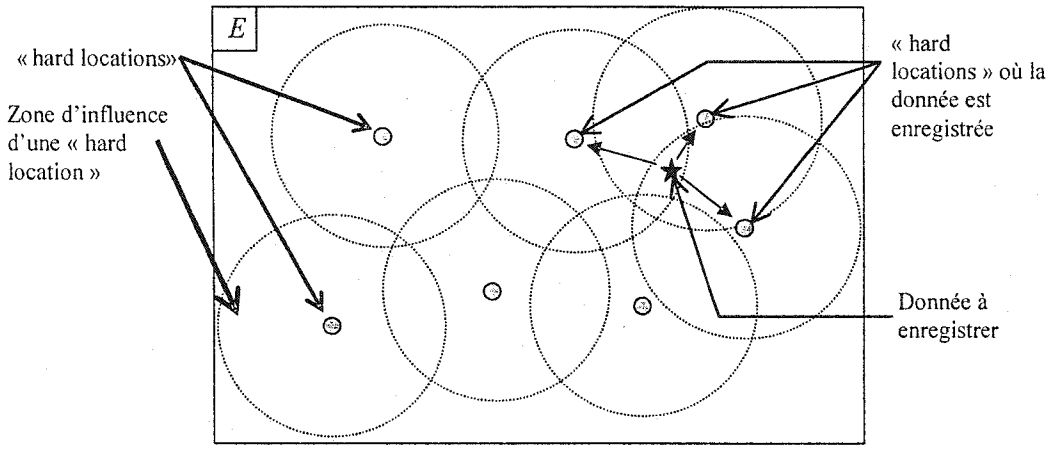


Figure IV.2 : Illustration de la phase d'enregistrement dans une mémoire de Kanerva

IV.3 RÉCUPÉRATION DES DONNÉES DE LA SDM

Le processus de récupération des données stockées dans la SDM se fait en 2 étapes : la première consiste à sélectionner les « hard locations » à lire, la seconde consiste à utiliser l'information qu'elles contiennent pour déterminer le vecteur de sortie.

IV.3.1 La sélection des « hard locations » lues

La sélection des « hard locations » à lire est le processus inverse de celui observé dans la phase d'enregistrement : les « hard locations » sélectionnées sont celles situées à k ou moins de k bits de la donnée présentée en entrée (Figure IV.3). Le nombre des « hard locations » sélectionnées, noté d, n'est pas connu a priori. Toutefois, dans un espace de

grande dimension, le nombre de « hard locations » sélectionnée est quasiment constant en raison de la distribution uniforme des « hard locations » dans l'espace des entrées.

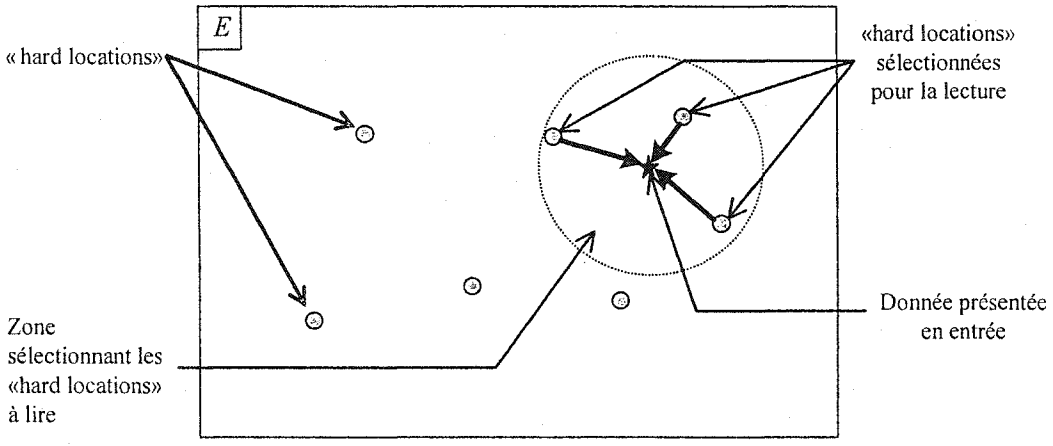


Figure IV.3 : Illustration de la phase de lecture dans une mémoire de Kanerva

On sélectionne ainsi un sous-ensemble de « hard locations » qui stockent les mots notés $\{W'_i\}_{i\in[1,d]}$.

IV.3.2 Le traitement des données sélectionnées

Le traitement des données sélectionnée s'effectue en 2 étapes : la première étape du traitement consiste à additionner vectoriellement les W'_i, la seconde étape consiste à seuiller[iv] cette somme pour obtenir un vecteur binaire en sortie.

[iv] Le seuil est fixé à zéro

Notons S le vecteur de sortie et S_j sa $j^{ème}$ composante. On a alors :

$$\forall j \in [1,d] \begin{cases} S_j = 0 \text{ si } \left(\sum_{i=1}^{d} W'_{ij} \right) \leq 0 \\ S_j = 1 \text{ si } \left(\sum_{i=1}^{d} W'_{ij} \right) > 0 \end{cases} \quad \text{(IV.3)}$$

IV.4 CONCLUSION

Le modèle développé par Kanerva a pour objectif de trouver le meilleur représentant correspondant à un vecteur d'entrée et possède en plus des caractéristiques permettant d'envisager une implémentation matérielle. Il semble donc parfaitement utilisable pour développer des CAM neuronales. En outre, Kanerva décrit toute la théorie du modèle permettant un bon dimensionnement du système dans son livre « Sparse Distributed Memory » [KAN88]. Le modèle des SDM, à l'instar des CMM binaires, nécessite l'utilisation de vecteurs d'entrée de grande dimension « sparses ». Si l'on veut intégrer des SDM dans une CAM neuronale, il faut les utiliser pour effectuer un prétraitement des vecteurs d'entrée.